AD-A257 212

DTIC
ELECTE
OCT 26 1992
S D
B

FINAL TECHNICAL REPORT

RAPID PROTOTYPING OF APPLICATION SPECIFIC SIGNAL PROCESSORS PROGRAM

Prepared For:

DARPA/ESTO
ATTN: Mr. Eliot D. Cohen
3701 North Fairfax Drive
Arlington, VA 22203-1714

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency of the U.S. Government.

Distribution Statement A. Approved for public release; distribution is unlimited.

Prepared By:

Texas Instruments
Integrated Circuity and Computers
Department
6550 Chase Oaks Blvd., M/S 8435
Plano, Texas 75023

October 1992

TEXAS INSTRUMENTS

92-27917

TABLE OF CONTENTS

Section	Title	Page
1.	INTRODUCTION	1
1.1	Study Approach	1
1.2	Work Summary	1
1.3	RASSP Vision	2
1.4	RASSP Design Approaches	2
1.5	RASSP Approach	3
1.6	RASSP Development Cycles	5
1.7	RASSP Program Requirements	5
1.8	Key Ingredients	6
1.9	Overview of Detailed Technical Discussion	7
2.	DESIGN LIBRARY	8
2.1	Target Platform Constraints	8
2.2	Signal Processor System Architecture	9
2.2.1	Functional Architecture Description	9
2.2.2	Architecture Implementation/Upgrades	11
2.2.3	Library Based Architecture Approach	12
2.3	Library Element Definitions	13
2.3.1	Standards	14
2.3.2	Test System	15
2.3.3	Software Library Elements	16
2.3.4	Hardware Library Elements	16
2.3.5	RASSP Library Views	18
2.4	Design Library Based Model Year Upgrades	20
2.4.1	Functional Element Design Examples	20
2.4.2	Standard MCM Library Element Examples	22
2.5	Summary of Issues and Recommendations	23
3.	DATA BASE AND DATA INTERCHANGE	26
3.1	RASSP System Architecture Overview	26
3.2	Summary of Related Work	27
3.2.1	Enterprise Integration	29
3.2.2	Information Management Standards	30
3.2.3	Generic Frameworks	31
3.2.4	Domain-Specific Frameworks	32
3.2.5	Data Base Management Standards	34
3.2.6	Distribution	38
3.2.7	Generic Interchange Formats	39
3.3	Issues/Recommendations	41
3.3.1	Issue: Object Model Harmonization Needed	41
3.3.2	Issue: Frameworks Harmonization Needed	43
3.3.3	Issue: The need for OODB Interface Standards	43
3.3.4	Issue: Who Will Win: OODBs or RDBs (Relational DBMSs)	44
3.3.5	Issue: The Need for Federated, Decentralized, Scalable Intra- and Inter-enterprise-wide Repositories (Data Base Systems)	44
3.3.6	Issue: Improvements to PDES/Express are Needed	44
3.3.7	Issue: Which Interchange Formats and Representations to Standardize?	44
3.3.8	Issue: What to put in the RASSP Library?	45
3.3.9	Issue: How to Bootstrap the RASSP Business?	45
3.4	Summary of Key Recommendations	45
4.	APPLICATION DESIGN SYSTEM	48
4.1	DSSA Software/DSHA Hardware Development	49
4.2	ASSP Application Design System	52
4.3	Design Integration and Validation	53
4.4	Extended DSSA Support Environment	54

TABLE OF CONTENTS (CONTINUED)

Section	Title	Page
4.5	Extended DSHA Support Environment	57
4.6	DSSAs/DSHAs	59
5.	SOFTWARE	62
5.1	Overview	62
5.2	Software Library	62
5.3	Software Engineering Environment	64
5.4	Summary	68
6.	COMPUTER AIDED DESIGN	71
6.1	RASSP Hardware CAE/CAD Investigation	71
6.2	CAD Process Flow Overview	77
6.2.1	System/Concept Level	77
6.2.2	Functional Design Level	77
6.2.3	Detailed Design Level	80
6.2 4	Design Verification Level	81
6.2.5	Manufacturing "Pre Fab" Interface Level	81
6.3	Design Information Modeling	82
6.4	RASSP Libraries	84
6.5	Tool Data Base Federation	86
6.6	Design Model Standards	86
6.7	RASSP CAD Recommendations and Summary	89
7.	MANUFACTURING	92
7.1	Foundry Interface/CAD	92
7.2	Manufacturing Overview	107
7.2.1	Quality/Reliability Plan	111
7.2.2	Production Test Needs	111
7.2.3	Test Solutions	111
7.2.4	Hardware Foundries - Test Interface	112
7.2.5	Test Technology Roadmap	112
7.2.6	Manufacturing Control	112
7.2.7	The Future of Computer Integrated Manufacturing	115
7.2.8	Next Generation Manufacturing System Contacts	116
7.2.9	Next Generation Manufacturing Issues	116
8.	BUSINESS PLAN AND PROGRAMATICS	119
8.1	Business Infrastructure and Teaming Environment	119
8.2	Program Plan	120
8.2.1	Standards Consortium	120
8.2.2	Technology Program	121
8.2.3	Tools and Library Program	121
8.2.4	Demonstration	122
8.3	Applications	122
8.3.1	Applications Criteria	122
8.3.2	Application Evaluation	124
9.	SUMMARY	136
9.1	Application Domain Selection Effort: ~1%Risk: Low	136
9.2	Data Representation Standards Selection Effort: ~5%Risk: Moderate	136
9.3	In-cycle Library Framework and CAx Tool Selection Development Effort: 30%Risk: High	136
9.4	Hardware and Software Standards Selection Effort: ~1%Risk: Low	137
9.5	Selection and Integration of the CAD/Case Libraries and Tools for Module Development Effort: 15%Risk: Low to Moderate	137

TABLE OF CONTENTS (CONTINUED)

Section	Title	Page
9.6	Select, Design, and Validate the Hardware/Software Modules to Support the Domain Specific Demonstration Effort: 15%Risk: Low to Moderate	137
9.7	Update and Demonstrate Seamless Interfaces to Flexible Foundries Effort: 5%Risk: Low to Moderate	137
9.8	System Demonstration Effort: 20%Risk: Moderate	137
9.9	Technology Transfer Effort: 5%Risk: Low to Moderate	138
Appendix A	Critical Issues and Recommendations	A-1
Appendix B	Key Standards	B-1
Appendix C	Bibliograpny	C-1
Appendix D	Distribution	D-1

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist Special
A-1

LIST OF ILLUSTRATIONS

Figure	Title	Page
1	RASSP Design System	3
2	Signal Processor Design Approaches	4
3	The RASSP Development Cycle	5
4	Signal Processor Architecture	10
5	Library Elements	14
6	Library Element Model Examples	18
7	Library Element Model Examples	19
8	Functional Element Design Examples	21
9	Standard MCM Library Element Examples	23
10	RASSP System Architecture	26
11	Summary of Related Work	28
12	Automated Mappings Should Exist Between These Three Views of Design Representations	40
13	Typical Breakdown of Major Activity Stages	48
14	Possible Top-level View of Formal System Design Process	50
15	Top-level View of Proposed RASSP Design System	52
16	Design Integration and Validation	55
17	Fundamental Components and Access Methods	56
18	Fundamental Components of Access Methods for DSHA Support	58
19	Three examples DSSA Architectures	60
20	Software Design Overview	62
21	Software Library	63
22	Five Levels of Software Process Maturity	65
23	Software Development - Today	67
24	Software Development - 1997	68
25	Software Development - 2002	69
26	Library Based Hardware CAD Concept	71
27	In Cycle and Out of Cycle RASSP CAD	72
28	RASSP Design Advisers/Tradeoff Analysis	75
29	Key RASSP Related Testability Trends	76
30	TI Design Flow	78
31	RASSP Design Flow	79
32	System Definition/Concept Design	79
33	Functional Design Level	80
34	Detailed Design Level	82
35	Design Verification Level	83
36	Manufacturing "Pre Fab" Interface	83
37	RASSP Design Information Modeling	84
38	RASSP Design Data Views	85
39	AnaVHDL Mixed Mode VHDL Environment	88
40	EDIF Information Model Views	90
41	Presentation Flow	92
42	MCM Merchant Foundry Status	93
43	Hardware Foundries	93
44	DSPIV Module	95
45	Foundry Interface System	96
46	MCC Summary of Tool Status - Known Tradeoff Analysis Tools	98
47	MCC Summary of Tool Status - Test Advisor Tools	99
48	MCC Packaging Synthesis System	100
49	Design Advisor Components	101
50	Foundry CAD System	103
51	Foundry CAD Demonstration for RASSP Program	105
52	Common Elements Associated with RASSP Program Manufacturing Environment	108
53	Tool Samples	109
54	Network of Foundries	110
55	Production Test Needs	112
56	Test Technology Roadmap	113
57	Foundry Manufacturing Control	113

LIST OF ILLUSTRATIONS (CONTINUED)

Figure	Title	Page
58	Foundry Data Automation System	114
59	Foundry Production Control System	115
60	Foundry Data Management	116
61	Teaming	119
62	Program Plan	121
63	Processing Flow	129
64	Typical Scanning E/O Sensor (Space Surveillance)	129
65	Generic Processor	130
66	Typical SAR Processor	130
67	Typical SBR Concept	131
68	ESM Processor	131
69	TMD Engagement Geometry	132
70	Top-Level Decision Tree for TMD	133
71	Range of Possible Roles in TMD	134
72	IRST Signal Processing	134

LIST OF TABLES

Table	Title	Page
1	Issues/Recommendations Summary	24
2	Enterprise Integration	29
3	Information Management Standards	30
4	Generic Frameworks	31
5	Domain Specific Frameworks	32
6	Data Base Management - Standards	35
7	Data Base Management - OODB Vendors	36
8	Data Base Management - R&D	37
9	Distribution	39
10	Genetic Interchange Formats	39
11	Harmonization Needed - Object Models and Frameworks	41
12	Data Base Management	42
13	Interchange Formats and Libraries (Generic)	42
14	Other Technologies Relevant to DSSAs/DSHAs	51
15	Software Engineering Environment Tools	70
16	Relevant Standards Versus Needs	85
17	CAD Activities	102
18	Required Capabilities for Foundry CAD	104
19	Foundry CAD Demonstration for RASSP Program	106
20	RASSP CAD Issues	107
21	North American Independent PCB Assemblers	111
22	Next Generation Manufacturing System Contacts	117
23	Next Generation Manufacturing Issues	117
24	TI Potential RASSP Application	125
25	TI Potential RASSP Application	125
26	TI Potential RASSP Application	126
27	TI Potential RASSP Application	126
28	Boeing Application Selection Criteria for a RASSP Demonstration System	127
29	Boeing Potential RASSP Application - Scoring	127
30	Boeing Potential RASSP Application - Scoring	127
31	Additional Airborne Platforms for RASSP Application	128
32	Additional Airborne Platforms for RASSP Application	135

LIST OF ACRONYMS

AHM	Anti-Helicopter Mine
API	Application Program Interface
APX	Array Processor Executive
ARTEWG	Ada Runtime Environment Work Group
ASIC	Application Specific Integrated Circuit
ATR	Automatic Target Recognition
BIST	Built-in Self Test
BIT	Built-in-Test
BSDL	Boundary Scan Description Languate
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CFI	DAD Framework Initiative
CFI	CAD Framework Initiative
CIFO	Catalogue of Interface Features and Options
CIM	Computer Integrated Manufacturing
CMM	Capability Maturity Model
CORBA	Common Object Broker
DARPA	Defense Advanced Research Project Agency
DoD	Department of Defense
DRAM	Dynamic Random Access Memory
DRC	Design Rule Checks
DSHA	Domain-Specific Hardware Architecture
DSP	Digital Signal Processor
DSSA	Domain-Specific Software Architecture
eCAD	electronic CAD
EDI	Electronic Data Interchange
ERC	Electrical Rule Check
FEWS	Future Early Warning System
HW	Hardware
I/O	Input/Output
IC	Integrated Circuit
IGES	Intermediate Graphics Exchange Specification
IIR	Imaging Infrared
IOBIDS	Input/Output Built-in-Test Description Specification
IOM	Information Object Modeler
IR	Infrared
IRDS	Information Resource Dictionary Systems
ISA	Instruction Set Architectures
JIOS	JIAWG Input/Output System
LDPS	Laser Drill and Patterning System
MANTECH	Manufacturing Technology
MCC	Microelectronics and Computer Technology Corporation
MCM	Multichip Module
MDP	Mission Display Processor
MES	Manufacturing Execution Systems
MGC	Mentor Graphics
NGCR	Next Generation Computer Resources
ODBC	Open Data Base Connection
ODMG	Object Data Management Group
OMG	Object Management Group
OODB	Object-Oriented Data Base
OSI	Open Systems Interconnection
PAP-E	PDES Application Protocols-Electronics
PCTE	Portable Common Tools Environment
PDES	Product Data Exchange using STEP
PDL	Performance Description Languate
POB	Persistent Object Base
POSC	Petrotechnical Open Software Corporation
POSIX	Portable OS Interface Processors
PWB	Printed Wiring Board
QML	Qualified Manufacturing Line

LIST OF ACRONYMS (CONTINUED)

RASSP	Rapid Prototyping of Application Specific Signal
RDA	Remote Data Access
SAG	SGL Access Group
SDAI	STEP Data Access Interface
SEI	Software Engineering Instruction
SES	Scientific & Engineering Software
SMT	Surface Mount Technology
SPPD	Signal Processor Packaging Design
SPX	Signal Processor Executive
STEP	Standard for the Exchange of Product
SW	Software
SWAP	Silicon Wafer Advanced Packaging Program
TEG	Test Element Group
TI	Texas Instruments
TMD	Theatre Missile Defense
TTS	Tank Thermal Sight
URDA	Ultra-Reliable Digital Avionics
VHDL	VHSIC Hardware Description Language
VIC	VME Interface Control
WAIS	Wide Area Information Server

1. INTRODUCTION

1.1 Study Approach

Texas Instruments approach to the RASSP study was to identify and select a development process consistent with the RASSP program methodology. This process consisted of:

1. Definition and capture of a domain specific signal processor system partitioned into interoperable portions with standard interfaces.

2. Support for model year upgrade of the signal processor system to improve system performance.

3. Rapid capture of commercial technology/best practice curves.

4. Reduced development and upgrade cycle time and risk and increased system affordability.

From this process vision, we derived the critical requirements of the Domain Specific Signal Processor and the development tools, databases, and standards that support the process.

We characterized the status and projected capabilities of these items by investigating the modularity and direction of recent TI and Boeing signal processing designs, as well as the effectiveness of the tools, standards, and manufacturing capabilities in place to execute the designs. We used our contacts with strategic vendors and standards organizations to project future capabilities.

We then developed recommended areas for emphasis and approaches to reduce risk in the RASSP implementation phase.

1.2 Work Summary

The TI/Boeing investigation encompassed the total signal processor life cycle - considering all the elements of concurrent engineering. We relied upon TI and Boeing's experience in signal processor efforts, and the each company's contacts with suppliers, customers, competitors, and standards organizations.

For example, at TI we have an internal department that develops advanced, modular computer and signal processor products to support internal TI system efforts, and to sell direct to other DoD system suppliers. TI has a large Engineering Services and Automation department that continually rates, selects, installs and supports CAE/CAD/CASE and, supported by a Manufacturing Automation department, interfaces these systems with our manufacturing control and enterprise integration activities. TI's Microelectronics Packaging Systems department provided the expertise for the foundry CIM and interface issues. TI offers custom manufacturing services to the commercial and DOD marketplace for PWB, SMT, MCM, and assembly needs. TI's Software Engineering department defined the software development process and supporting tools investigation. TI is a disciple of the SEI software process management techniques. We have major efforts underway to install tools to support this process and university funding to enhance these capabilities. We use our Corporate Research Computer Science Laboratory to look at advanced data base and data base management techniques.

Each of these organizations used direct contact (RASSP discussions) and indirect contact (a part of the on-going process of performing their business) with suppliers, standards organizations, and universities to

gather data for the study. Examples of these contacts include the following:

Suppliers

- Mentor Graphics - design frameworks, hardware modeling and simulation
- Cadre - software capture and validation
- Scientific & Engineering Software (SES) - system modeling and simulation

Universities

- University of Cincinnati - performance description and modeling
- Carnegie Mellon - systems analysis software

Standards Organizations

- IEEE DASS - VHDL
- EIA - EDIF electronic data interchange
- CAD Framework Initiative - design data representation standards

1.3 RASSP Vision

Figure 1 illustrates the infrastructure for our vision of the seamless development environment for RASSP. We envision a design library based signal processor approach similar to the standard cell approach used by the silicon ASIC industry. A federated set of libraries will contain the design representations and model views required by each member of the integrated product development team.

The system design team library contains the modular, re-usable software (SW) and hardware (HW) elements and physical, functional, and performance models of the elements. The tools support analysis and requirements allocation, configuration of the modules to meet design requirements, and validation of the design.

The software design team generates the reusable SW modules that feed the library. The tools support software development, documentation and validation of these modules.

The hardware design team develops the processors and interface modules for the signal processor domain. This team's library contains the data elements for detailed design including parts descriptions and design rules for PWBs, MCMs, ASICs, etc., and the tools that support these designs.

The manufacturing/test team translates the module and system design representation into the data that drives these manufacturing and test systems.

Standards assure that each design team member has immediate access to the data representation (or model view) needed to perform his/her task.

1.4 RASSP Design Approaches

The conventional "waterfall" design process (part a of Figure 2) will not meet the cycle time goals of the RASSP vision, even with the

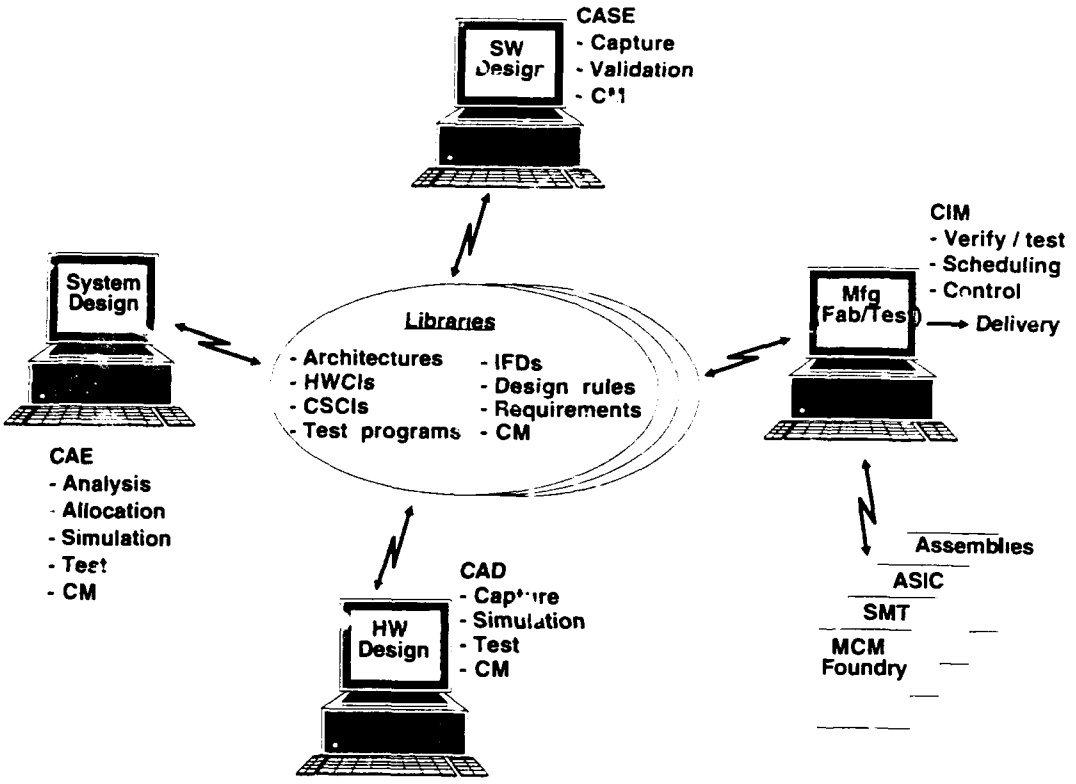


Figure 1
RASSP Design System

availability of the modern tool sets supporting each activity. The "cycles of learning" are too long and design iteration costs are too high.

The effective near term goal is to develop and maintain a library of signal processor elements for a particular domain. These element designs are performed and detailed "out-of-cycle" with the critical path of the prototype program. The "in-cycle" design selects and configures the library elements to meet the system requirements. The library elements are complete, validated processor and interface modules configured for MCMs or PWBs, plus reusable software modules for operating systems, communication, and application configuration items.

In the longer term, we believe that synthesis techniques will permit additional gains in cycle time reduction. The design paradigm is similar: but in this view, the design synthesizers themselves contain the building blocks, and thus library updates are essentially tool updates.

Figure 2 illustrates the differences in these design approaches.

1.5 RASSP Approach

The near term approach (Figure 2, part b) reduces cycle time by increasing reuse of pre-validate SW and HW building blocks. Definition of the level of these blocks and how they interface sets the apparent complexity of the in-cycle design task. Too high a level will reduce needed domain specific flexibility, and make upgrades more complex.

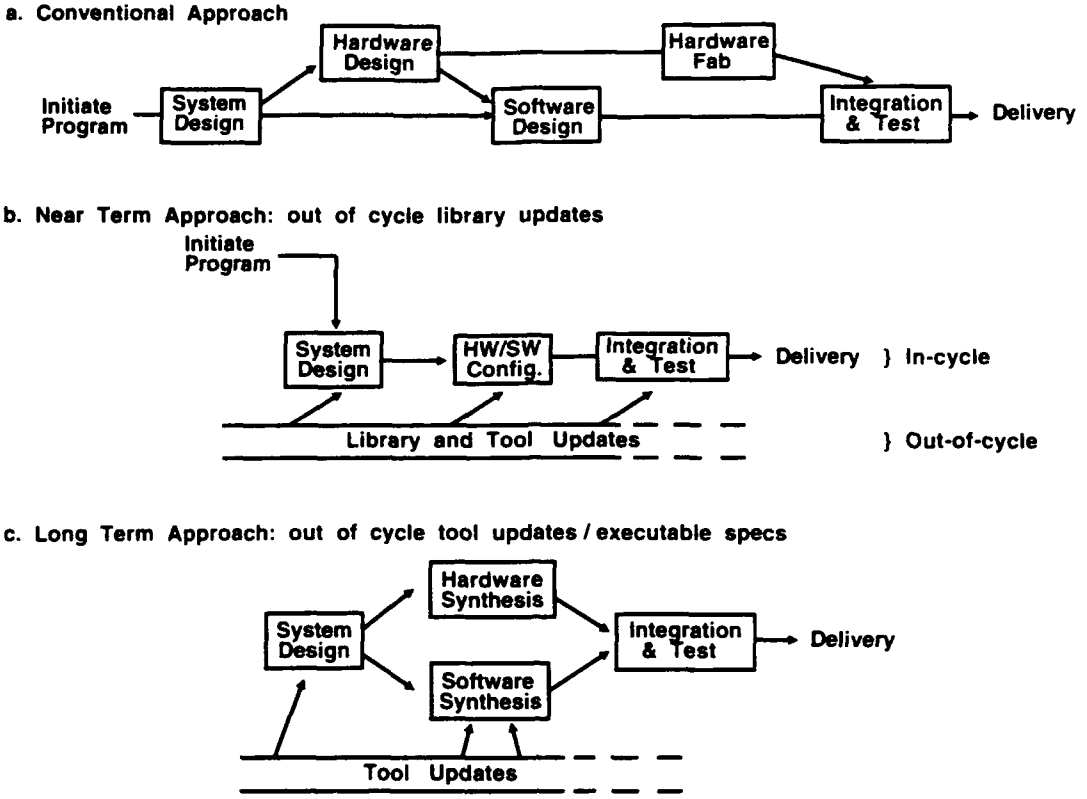


Figure 2
Signal Processor Design Approaches

The model views of the building blocks delivered to the system designers must be at the higher level of abstraction to permit adequate "real time" equivalent simulation at the system level to achieve the "design correctness by simulation" goal. The more abstract models must be validated against the more complete, detailed models.

The approach must reduce the non-value-added activities of the design cycle. The approach is to deploy a seamless environment via common data representation standards at the level needed by each development participant.

The approach does require incentive for out-of-cycle library maintenance and upgrades. We believe the industry restructuring, and progress in design tools for the detailed design of ASICs, MCMs, and PWBs, will continue to improve library build capabilities.

In the long term (Figure 2 part c.), design synthesis and expert design advisor tools will permit even shorter cycles by lessening detailed design tasks via executable specifications. The detailed design is thus "correct-by-design". Common, object-oriented data bases for all design and manufacturing views will be available to further reduce inefficiencies. Primitive versions of such tools are available now.

There will remain a significant, continuous, "out-of-cycle" effort to upgrade the tools to accommodate technology improvements.

1.6 RASSP Development Cycles

Figure 3 illustrates the infrastructure's impact on the development cycle. Library development is a continuous process, and updates are driven by the technology improvement curves of components applicable to the domain. The prototype development cycle is reduced to configuration of the library elements into the system.

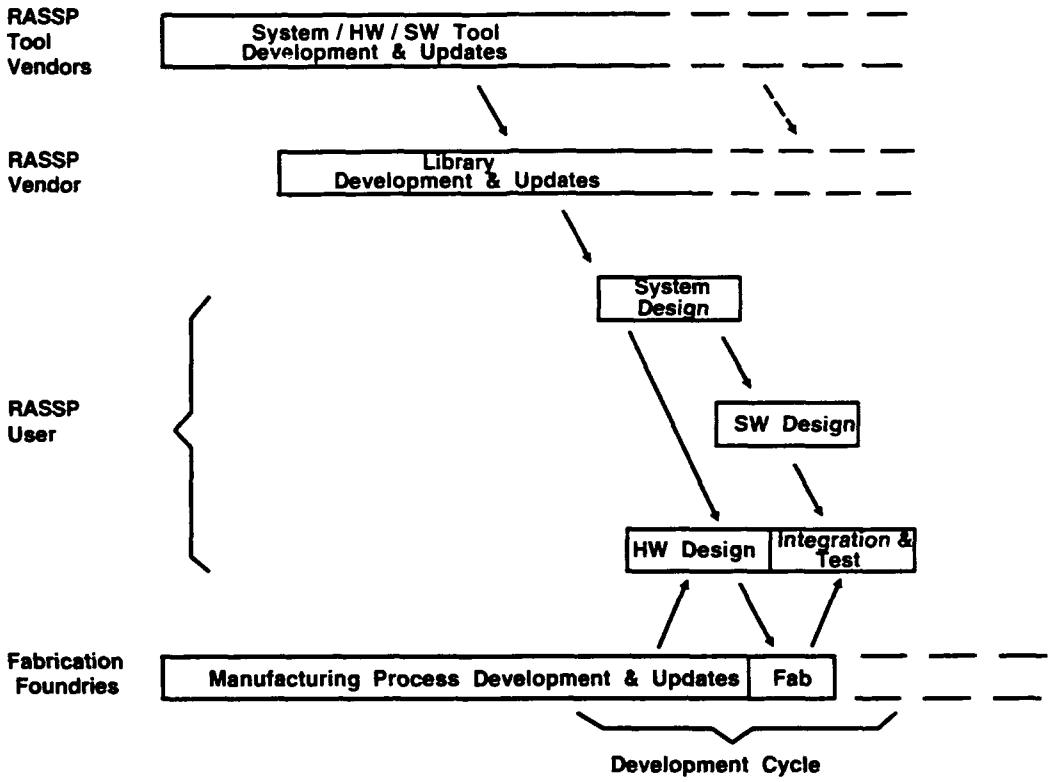


Figure 3
The RASSP Development Cycle

The cycle time of model year updates is set by the cycle time to develop the processor, interface and/or software module affected by the update, plus reconfiguration of the new element into the system. Thus, both in-cycle and out-of-cycle tool sets are important.

The domain specific structure minimizes the breadth of the impact of the changes. Thus, the modularity and standard HW/SW interfaces of the design are key cycle time drivers.

TI and Delco have just established such an infrastructure for automotive signal processing applications. The modularity and level (complexity) of the library elements are key to cycle time reduction whether we are building a one-to-three IC automotive signal processor, or a 10-to-20 board ATR signal processor.

1.7 RASSP Program Requirements

The RASSP implementation phase must concentrate on cycle time and design integrity (quality). Experience has shown that we improve productivity, affordability, and design quality with a cycle time emphasis.

Productivity emphasis usually improves a piece of the process, but can impact the overall goal.

The requirements of the RASSP program are:

Concentrate on complete, effective tools at the library module level and above. Industry progress on ASIC and board level tools is excellent. The help needed here is in design representation standards and model availability. The primary goal of RASSP should be to integrate available tools into an effective, open, seamless environment. DoD signal processing suppliers have similar in-house capabilities, but the environment is unique to that single organization.

Set stretch goals. We recommend a 25 percent reduction per year in average cycle time. Define the process and benchmark each piece of the process as you integrate it into the environment. This will require development of "standard" test cases for each tool or piece of the process.

Permit re-engineering of the process as required to achieve cycle time goals. Allow learning via the benchmark exercises.

Provide incentives for continuous improvement of the libraries and the design tools. The RASSP environment must be an "open" system. The tools, infrastructure, and design/foundry capabilities should be available industry wide. Much of the affordability gain comes from reuse.

1.8 Key Ingredients

Library element definition including function, structure, and standards is key to minimum cycle and low risk upgrade. For example, the segmentation of operating system, communication routines, and algorithmic execution functions will determine the impact of a processor upgrade on the overall system. The definition must be complete. All model views must be in the library. However, do not over standardize. The greatest improvements may come, for example, by changing a processor to a completely new instruction set architecture, or maybe the same ISA with radically redefined memory or I/O interface. If we correctly capture the domain elements, system trades will determine the viability of inserting the new components.

The need for data representation standards is endorsed by industry and there are many on-going standards efforts. Availability of a controlled, complete, accepted standard to meet RASSP timelines is at risk due to the time standardization efforts require. An open RASSP working group can select the best from these standards and adapt procedural overlays to these standards as required to accommodate incompleteness - with a goal of moving to the final standard when available.

System design improvements are critical. Current art is for this case only, unique to each signal processor application, and systems models are quickly outdated by design changes. Emerging tools can be adapted to RASSP on a timely basis. Domain constraints and the level of the module library makes this possible.

Excellent progress in software process and supporting CASE tools is occurring in industry. By institutionalizing reuse on the RASSP efforts, we can reduce the cycle time and risk impact of software. Software processes must be a major part of the RASSP infrastructure or gains will be minimal.

CAD systems and the models that support them at the module level and above must be improved to permit affordable validation via simulation and assure that the higher abstraction models are a correct and complete abstraction of the lower level as-built modules. Performance modeling must be addressed.

Tools for design of the ASICs, MCMs, and PWB module designs are progressing well. The RASSP effort is to select, integrate and conform inputs, outputs, and models to data representation standards. (This effort is RASSP proceduralized where required.)

RASSP can help the proprietary model availability program by supporting the encryption techniques becoming available in some CAD tools.

Global competitiveness is forcing rapid movement to flexible factories. DoD is a necessary participant in start-up of foundries for newer technologies (MCMs). The key is to ensure that these efforts remain funded and that multiple open foundries are integrated into the RASSP environment.

1.9 Overview of Detailed Technical Discussion

The following sections of this report detail the requirements and identify the risks for each element supporting the vision of the RASSP infrastructure.

Section 2 discusses the modularity (partitioning) of a domain specific architecture and the library elements to support this architecture.

Section 3 discusses the issues associated with data bases, frameworks and data representation interfaces required to provide a seamless RASSP development environment.

Section 4 details requirements, issues, and approaches to the signal processing system design - requirements capture and validation, systems analyses and trades, and systems simulation and validation.

Section 5 discusses the details and issues of developing reusable software and library elements for use by the systems designer.

Section 6 discusses the issues associated with the hardware CAD used to develop the hardware library modules, and support system level analyses and simulation/validation efforts.

Section 7 presents the requirements for interfacing with manufacturing (foundries) and issues with foundry CAD, CIM and test.

Section 8 discusses the business issues of the RASSP environment, including programmatic concerns, the interdisciplinary teaming required for successful execution, and the criteria for selecting the RASSP demonstration system(s).

Section 9 presents a summary of the RASSP study, discussing required capabilities that will be developed independently of the RASSP implementation program, and highlighting those capabilities that RASSP must concentrate on to provide an integrated RASSP infrastructure.

2. DESIGN LIBRARY

Implementing the RASSP program using a design library concept provides a rapid prototyping cycle capability for developing new system designs and implementing system upgrades. The key concept is that new technology is inserted into a library of functional element designs out-of-cycle with the development of signal processing systems. These new technology functional elements then provide the basis for new system designs or for upgrades to existing systems. By focusing the system development efforts on system design using pre-developed library elements, the system development cycle time is reduced.

The level at which library elements are defined is key to meeting the variety of design constraints within an application specific domain. Defining a library of elements solely at the module level will not provide a cost effective solution for rapid development of signal processors meeting the variety of constraints and requirements within an application specific signal processing domain such as ATR processing. Many of the design constraints are driven by physical environment drivers (such as volume, weight, and power allocations and cooling capacity availability). Other programmatic drivers, however, can be as important in influencing a signal processor design. The relative emphasis on operational reliability, maintainability, spares inventory minimization, functionality, performance, standards, programmability, and system cost also drive the design. By defining library elements at a variety of levels (module, multichip module, and functional design element), library based signal processor developments can accommodate the variety of target platform specific constraints within an application specific signal processing domain.

In defining a library of functional elements, the following requirements, constraints, and characteristics must be considered:

- Target platform constraints
- System architecture features and upgradability
- Hardware library element constraints and requirements
- Software library constraints and requirements.

These constraints and requirements are driven by the characteristics of the selected application specific signal processor domain. The following discussions focus on Automatic Target acquisition, tracking, and Recognition (ATR) domain signal processing. However, much of the discussion is generic in nature and is appropriate for any application specific domain.

2.1 Target Platform Constraints

Within the ATR signal processing domain, ATR processing is applied to missile, ground weapons system, and airborne platform environments. Each of these environments has constraints and requirements that tend to provide different design drivers for each application. The primary difference among the platform types is the degree of emphasis on the relative importance and severity of the associated constraints imposed on the system and module design. The result is often a considerable variation in signal processor architectures across this span of applications. A common signal processor architecture or module approach will not satisfy the needs of these applications. However, system designs using a design library concept based on appropriate definitions of functional library elements can meet the needs of the ATR specific signal processing domain across most of the ATR application environments. A multi-level library approach of module, MCM, and functional design elements provides a variety of library elements that can be assembled into module and system designs of varying form-factors,

packaging technology, and scalable performance/functionality that will allow rapid cycle signal processor developments specific to the target platform constraints.

Missiles are usually under tight size, weight, power, and cost constraints. The missile environment often produces odd form-factor requirements with little commonality across different applications, and usually has severe cooling capability constraints. These constraints and requirements often drive missile oriented signal processor designs to conduction cooled, odd form-factor (round, half-moon, long-skinny rectangular), lowest cost (with adequate performance) implementations with a greater emphasis on shelf life to fire reliability than operational maintainability.

Ground based systems (e.g., launchers, tank fire control systems) tend to be slightly less constrained than missile oriented systems for size, power, and recurring cost, but still have severe cooling system capability constraints (usually no external coolant, and possibly no airflow). Ground equipment space requirements can often be accommodated with standard chassis form-factors (VME, SEM-E, etc.) without the need for the unusual shapes often required in the missile applications. Ground based signal processors tend to be designed for high operational reliability and maintainability using conduction cooled, standard form factor (or at least more common form-factor), low cost (with adequate performance) implementations.

Airborne platforms (pods and bays, rotor or fixed wing) emphasize operational reliability, maintainability, high performance/weight ratios. They have a high degree of commonality across functions with looser constraints on size, power, and recurring cost than missile or ground applications. Airborne platforms also provide external cooling capability (air cooled chassis, liquid flow-through chassis or modules). These requirements and constraints tend to emphasize designs with high performance, functionality, and flexibility for multiple applications over individual module cost, size, power, or performance optimization for a specific application. Airborne signal processor designs often implement high performance, highly programmable signal processor elements in standard form-factors, allowing conduction cooled and liquid flow-through module cooling approaches.

2.2 Signal Processor System Architecture

While signal processing systems within the ATR processing domain may closely resemble one another functionally, there are very distinct differences among systems due to physical configurations, sensor types, and system applications. These differences are driven by the target platform constraints, programmatic constraints, and performance/functionality requirements. The range of physical implementations of a signal processor functional architecture needs to be addressed when defining the library elements for a library based signal processor development approach.

2.2.1 Functional Architecture Description

The typical functional representation of a signal processing system, as shown in Figure 4, consists of a sensor (possibly multiple sensors), A/D and analog-to-digital functions associated with the sensor(s), a digital front end, application specific signal processing, and data and control processing.

The analog-to-digital and A/D functions are very sensor specific. These functions usually include the analog-to-digital conversion of sensor data signals and digital-to-analog conversion of platform and sensor

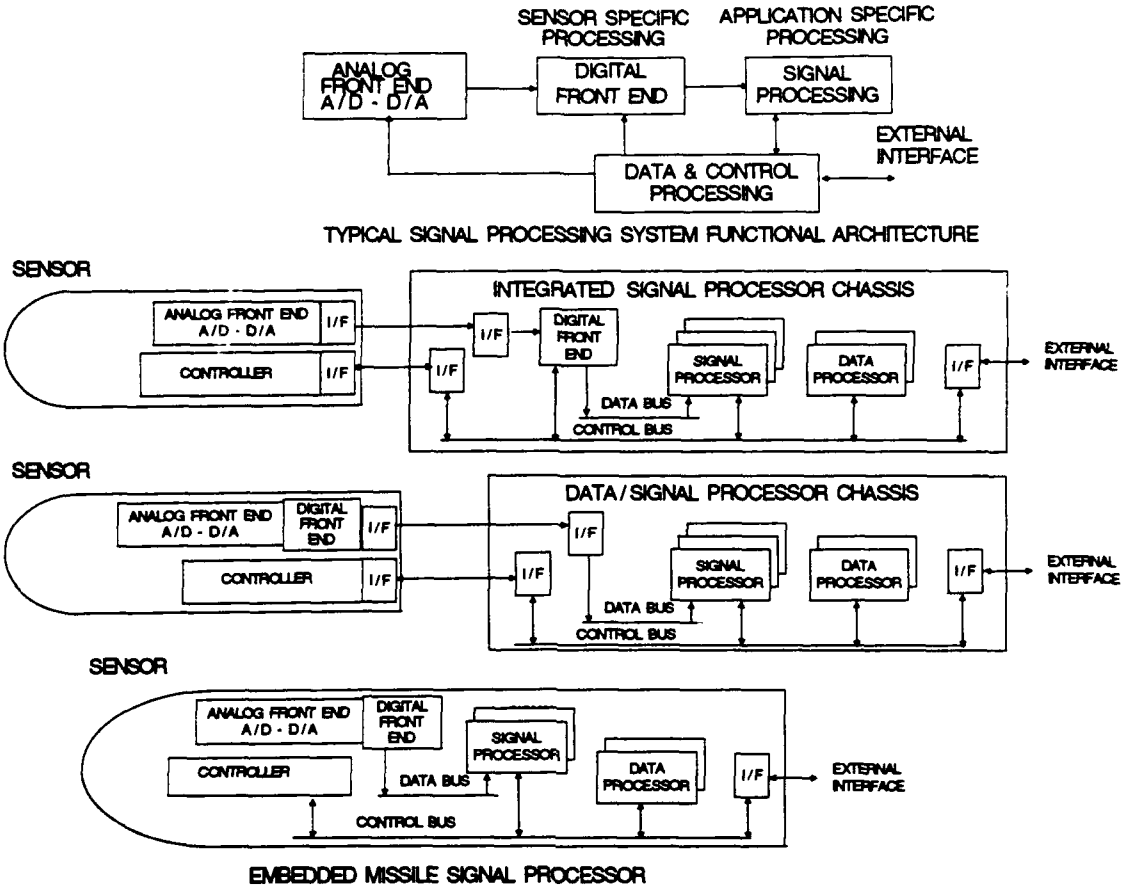


Figure 4
Signal Processor Architecture

control signals. They often comprehend sensor specific control functions. Sensor data may include position or rate sensors associated with primary sensor pointing and platform status (airspeed, orientation, altitude), as well as data from the primary data gathering sensor(s) (radar, IIR). Control signals may include sensor pointing commands, timing controls, or other controls for the sensor. Control functions include sensor data sample timing generation, sensor pointing stabilization, and sensor movement control. The analog functions are usually included in the sensor electronics with the sensor, and are often embedded as close to the sensor elements (IR detectors, focal plane array elements, radar transceiver units, etc.) and platform sensors/controls (rate gyros, position sensors, servo-motors, gimbal controls, etc.) as possible. Nearness to the sensor elements (often combined into one assembly) and platform controls minimizes the introduction of noise into the analog signals by the remainder of the system.

Digital front end processing typically consists of sensor specific processing required to provide data in the proper digital format, resolution, and signal integrity. FLIR sensor data, for example, usually requires gain and level correction, bad pixel substitution, and data formatting before the application signal processing can be performed. Radar data is usually formatted into digital pulse reports prior to application signal processing. Digital front end processing may be performed by function specific hardware (hardwired arithmetic

functions performing a specific non-reprogrammable task), limited programmability digital processors (such as some FFT processors), or programmable processors (DSPs, micro-programmable vector processors). The selection is dependent on the physical constraints and functional performance requirements.

Application specific signal processing performs the computationally intensive processing associated with the application functions. These functions consist of a collection of algorithms and a processing flow specific to the approach used in solving the application problem with the selected sensor suite. For an ATR system using a FLIR sensor, these algorithms may include spatial filters, image enhancement, feature detection, and other pixel intensive or data dependent processes. Application specific signal processing is often performed by high performance programmable digital signal processor modules or components. Programmability is important because of the need to meet changing requirements and to allow the use of common core signal processing hardware across multiple system developments. Ease of programmability and availability of a highly productive software development support environment is often traded-off against the need for high computational rate performance on data intensive processing.

Most of the decision intensive and system control processing is usually performed by a data and control processing function. This function usually provides overall system control and orchestrates the application specific processing flow. Data and control processing provides control of the application specific signal processing elements, the digital front end processing, the sensor, and possibly the entire system. For example, control fin and thruster commands, sensor gimbal pointing commands, and targeting information to a gunner may be provided by the data and control processing function in a missile signal processor system. The data and control processing function usually comprises the majority of the system design, and is often very software intensive. Computational performance is often traded-off against the need for a high level of programmability, highly productive programming support environment/tools, and maintainability of code. Data and control processing is usually performed by highly programmable scalar processors (RISC or CISC) with extensive programming support environments for standard high order programming languages (Ada, C, C++).

2.2.2 Architecture Implementation/Upgrades

The functional architecture may be implemented in several different physical configurations. These configurations are often dictated by programmatic or physical constraints and requirements. Figure 4 shows several possibilities for the physical implementation of a typical signal processor functional architecture.

While the sensor analog-to-digital and A/D functions are usually contained within the sensor electronics, digital front end processing and application specific signal processing may be grouped together, either in a common integrated signal processor chassis or in the sensor electronics, or may be separated between the sensor electronics and a signal processor chassis. As shown in the Figure 4, the data processors are often grouped with the application specific signal processors. However, there is no rule that data processors and signal processors have to be grouped together, and some systems may have data and signal processors physically separated.

The physical implementations shown in Figure 4 could represent configurations for different system applications. However, this figure could also represent a progression of system upgrades through the life of a particular program. The initial physical implementation is

consistent with the approach often taken for early concept demonstration programs (especially missile system developments). The sensor and sensor electronics are packaged in a prototype final form-factor and the processor is implemented for flexibility and performance, without a hard form-factor requirement. As the program transitions to a technology demonstration phase, inclusion of the sensor specific processing (the digital front end) in the sensor electronics may be desirable for programmatic or technical reasons, if the algorithms and design are proven and stable. The remainder of the processor may be re-implemented to meet some other new physical configuration constraints or requirements (such as flight testing in a pod vs lab or ground based testing). However, the signal processor may not be transitioned to the final form-factor due to a need for continued flexibility as requirements may continue to be perturbed from test results, and because the development costs may not be supportable in this phase of the program. The final configuration then (in this example) implements all processing within the sensor electronics.

Performance upgrades may also be necessary throughout the life of the program. Performance upgrades may be coincident with configuration upgrades or may take place independently. They can consist of hardware (component or module) technology upgrades, combined hardware and software upgrades, or software only upgrades.

In other systems (e.g., F-22 CIP, Comanche signal processor), the initial configuration (Integrated Signal Processor Chassis in Figure 4) is often close to the final configuration. In these cases, the initial implementation may not meet the final form-factor or performance requirements, but the physical partitioning is essentially the same as in the final system implementation. Upgrades tend to be limited to hardware and software enhancements within the functional elements (digital front end, application specific processing, and data and control processing) rather than evolutions of the physical configuration of the architecture. Module level configuration upgrades are usually planned to occur as the program transitions through the different program phases to ultimately provide the required performance within the application system's production form-factor and environmental requirements/constraints. Again, upgrades may take place as configuration upgrades (form-factor re-implementations) or performance upgrades (involving hardware, software, or both).

2.2.3 Library Based Architecture Approach

In defining a library of elements for signal processor design, selection of a set of standards for hardware and software interfaces is very important. Limiting the allowed number of different hardware and software interfaces minimizes the costs of system upgrades and library development. However, the selection of standard interfaces between processing functions is impacted by the physical partitioning, the communications requirements, and the physical and functional constraints of the system implementation.

As shown in Figure 4, the physical interconnection between two functions (such as the digital front end and the application specific signal processing) may differ across different systems, or may change with the program phase transitions. Standard hardware interfaces allow a rapid transition of an existing design to a different physical interconnection, either for an upgrade or for hardware design re-use. Use of standard OS/hardware and OS/application SW interfaces reduces the impact to the software when transitioning to a different physical interconnection for an architectural upgrade and maximizes software re-use across different system applications.

Hardware interface standards include the functional protocol and the physical interconnect. Physical interconnects may include fiber optic, wire cable/harness, or electrical backplane constructions. Fiber optic or wire interfaces will generally provide the communications interconnection between the signal processor and the external world (other subsystems, control functions, displays, operator interfaces, etc.), and high speed data interfaces between sensors and the signal processor (FLIR, radar, etc.). Backplanes, fiber optic, or wire interconnections may be used internally to the signal processor for module-to-module communications. The library functional interface standards need to include the backplane bus types (VME, VSB, PiBus, CIP Data Network, Futurebus+) and external communication interface standards (1553, High Speed Data Bus) that are commonly used, or projected to come into widespread use, in DoD programs. The current functional and physical standards for test and maintenance interfaces (JTAG, TMBus, VTMBus) also need to be included in the library.

Standardization of software interfaces is more a case of definition than selection. There has not been enough consensus for the standardization of software interfaces, especially for distributed communication, fault tolerance, and security standards. Much of the definition of OS/application SW interface standards may be provided by the POSIX effort. OS/hardware interface standards, however, have not been well addressed. The JIAWG Input/Output System (JIOS), defined as part of the IOBIDS program, demonstrated the effectiveness of a standard OS/hardware interface for PiBus controllers and driver software. OS software on different vendor modules was able to interface to the PiBus using a common JIOS software interface. Enhancements to the POSIX standards may be required to meet typical embedded signal processor real-time requirements. Additional OS/hardware interface software standards will need to be developed, and communication standards for inter-processor and external system communication will need to be provided as part of the library standards.

By providing a limited but comprehensive set of interface standards for implementation of the hardware and software library elements, the RASSP library concept will facilitate rapid cycle upgrades across many physical system partitionings and applications.

2.3 Library Element Definitions

Figure 5 shows that the deliverable elements of a system can be decomposed into operational hardware and software subsets and test related hardware and software. The hardware can be further broken down into subsystem, module, and lower levels of physical and functional elements. Software likewise can be divided into application software, OS software, and lower levels of software subsets and software modules. Test support equipment, hardware, and software are required for module unit test, system integration and test, and maintenance of a deliverable system. However, the test elements are not quite as easy to break out into a hierarchy of separately identifiable (or deliverable) pieces. Many of the test elements are often distributed with and embedded in operational hardware and software, or are closely associated with test of a specific piece of hardware or software.

System upgrades are accomplished through production and assembly of modules (hardware or software) that provide a new or improved set of characteristics to the existing system. A library based upgrade approach with modularity at the appropriate levels of hardware and software will provide a rapid and cost effective signal processor prototyping capability. To facilitate design re-use and ease system upgrade integration, interface and design standards must be applied to the development of all library elements.

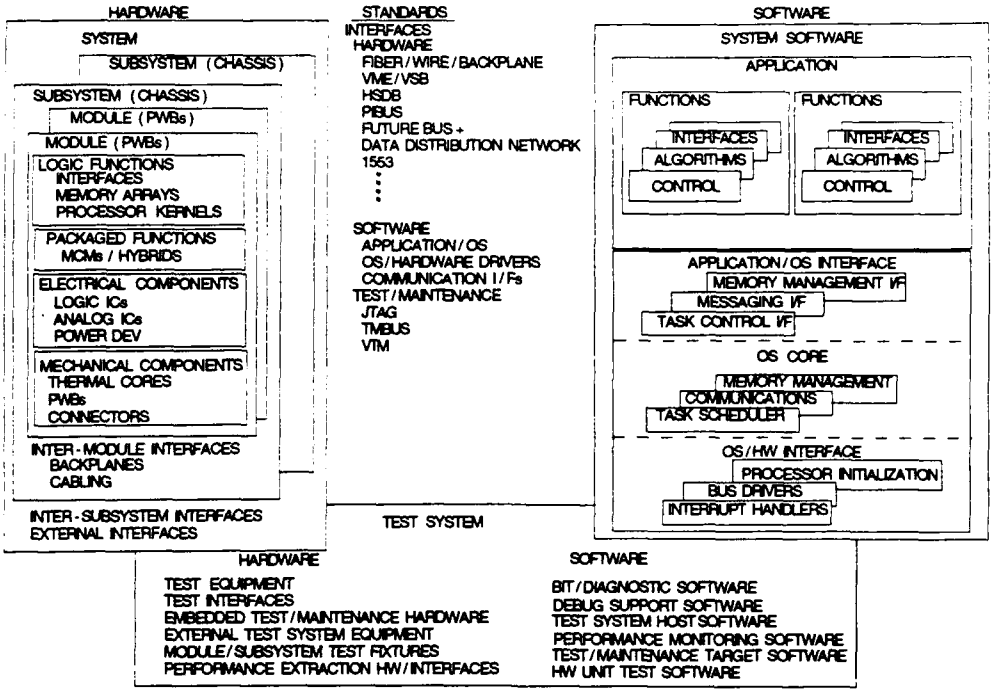


Figure 5
Library Elements

2.3.1 Standards

The key to an effective library based upgrade approach is to focus on hardware and software interface standards so that changes in ISA, system architecture, or hardware elements have limited affect on existing application software, the system operation, and hardware/software inter-operability. While some standards are necessary, the RASSP library approach should not insist on enforcement of single standards (interfaces, ISAs, form-factors, or even programming languages) for the operational aspects of the library elements. Test and maintenance, and performance monitoring interfaces, however, do need strict enforcement of a limited number of standards.

Operational Functions

In past common processor development efforts, common standards definitions have encompassed form-factors, buses, processor Instruction Set Architectures (ISAs), and programming languages that must be supported for the standard ISAs. These have often been enforced as single standards in each area (e.g., single slot SEM-E, PiBus, 16 bit 1750A ISA, 32 bit i960MX ISA, Ada). This approach can result in inter-operability across similar systems, a limited inventory of module types (for a particular system), and common software tools. However, if applied too stringently, these standards can be so restrictive that technology improvements cannot be easily inserted into these systems. Limitation to a specific ISA, for example, only allows system upgrades to take advantage of enhancements to the selected standard ISA component. Unfortunately, it is most often the case that the latest processor technology is implemented in a different ISA than the selected standard. It also seems that processors implementing whatever ISA has

been selected as the standard never match the technology improvement curve followed by new processor offerings. Standardizing software, hardware, and hardware/software interfaces minimizes application and OS software redesign, eliminates upgrade introduced interface incompatibilities, and reduces hardware and software rework in other areas of the system. With out-of-cycle development of library elements designed to meet the library supported interface standards, the system designer is allowed to capitalize on the industry-wide technology improvement curve, rather than only following the improvement curve of a single ISA standard product.

Enforcement of strict physical and interface standards (e.g., SEM-E form factors only, PiBus only), also impacts design re-use for system configuration upgrades and across new system developments. A SEM-E processor module for an avionics embedded computer system will not fit in a 6-inch diameter missile body. Inclusion of functional design elements in the library allows a design to be ported across multiple form factors, packaging technologies, physical interconnect technologies, and system architectures.

The RASSP library approach should minimize the number of interface, ISA, and programming language standards used by the library elements. However, trade-offs rather than edicts should determine when the out-of-cycle library element developments should be allowed to deviate from the current standards to take advantage of the latest technology.

Test and Maintenance

A single standard for the component level test interface (e.g., JTAG) is mandatory to minimize module design complexity. A single standard for the module level test interface (e.g., TMBus) is desirable, but a limited number of standards can be supported (e.g., JIAWG TMBus and SAVA VTMBus).

Performance Monitor Functions

Performance monitoring interface standards should also be standardized to allow development of common external test equipment and test systems. Performance monitoring involves the collection of data (algorithm computations, control information, and timing) needed to determine that the system is operating correctly (e.g., correctly discriminates between targets and clutter, meets the operational time line, properly points the sensor) under test and known operational conditions. Often the data and information must be collected and analyzed after the fact to determine whether or not the system performed correctly. Performance monitoring functions currently are often embedded in system software, may use a combination of test and operational interfaces (usually resulting in limited monitoring capability), or a special hardware interface is implemented. Separate interface standards need to be defined for performance monitoring because the currently defined standard test interfaces and most operational interfaces usually do not have the required functionality or bandwidth to support the level of data collection needed in most systems.

2.3.2 Test System

Use of a library of standard test system elements (interface hardware and software, host system equipment) will minimize test equipment/ software development and the time spent in system integration for new system developments and upgrades to existing systems. External test system hardware and software elements may only be represented in the library as documented designs for fabrication and delivery with a signal processor system. Embedded test hardware and software functions and

components will be accessible for inclusion in the functional hardware and software design of the signal processing elements. These may include functional design elements such as fault logging and test interface hardware and software, fault reporting hardware and software, or target resident BIT and diagnostic software. By developing test system design library elements, upgrades to the test system hardware and software (either embedded or external) can be made as necessary to support signal processor operational system upgrades.

2.3.3 Software Library Elements

As shown in Figure 5, signal processor system software can be partitioned into application and operating system software. Further partitioning of the software into lower level functions provides the basis for designing re-usable software library elements. Implementation of the signal processor system software as re-usable software library elements is critical to reducing software development and upgrade cycle times. Implementation of the library elements to software interface standards is the key to minimizing the impact to software of system (software or hardware) upgrades.

Within the application software, functions must be partitioned carefully for implementation as re-usable software elements. In general, communications or interface functions should be separated from control and algorithm computational software. Algorithm upgrades are the most common software upgrade, and occur during system development, at transitions between program phases, and as product improvements after delivery of a system. Algorithm upgrades can consist of algorithmic or timing performance enhancements to a basic algorithm implementation, or as additional algorithms to implement new functions. Communication and control functions may be upgraded as a system hardware architecture evolves, because functions are added to a system, or as a result of hardware performance upgrades. Designing for re-use at these levels minimizes the impact of algorithm implementation upgrades on control and communication software and vice versa.

For the operating system software, the most desirable approach is to base the OS library elements on a set of common operating system functions. The OS design will be tailored, by selecting from the design library elements as needed, to meet a specific application's needs. A limited selection of ISAs and programming languages will be supported for implementation of a tailored OS design. Separation of the OS into application/OS interfaces, OS/hardware interfaces, and OS core functions isolates the impact of many hardware or software upgrades to a limited set of functions. For example, for a hardware upgrade that changes the physical interconnect and interface logic between processors, but produces no change in functionality, the impact may be limited to changes in the OS/hardware interface drivers. The OS core, application/OS interface, and application functions can be isolated well enough from the hardware to see no impact for many types of upgrades.

A re-usable library of software elements approach does offer the possibility of performance degradation to meet a design for re-use requirement. This may require a trade-off of performance versus design for re-use benefits. The RASSP library will be based on the design for re-use approach, but will also accommodate performance optimized implementations.

2.3.4 Hardware Library Elements

As Figure 5 shows, signal processing system hardware can be decomposed into subsystems (chassis assemblies), the chassis components (modules,

backplanes, wiring) and sub-module components (devices, MCM assemblies, and functional designs).

The system and the lower level subsystems may be modeled in the RASSP library for system performance analyses, or only documented as an assembly for fabrication. The representation depends on the needs of the system designer and the sophistication of the tools available to support the various forms of representation. In general, the system and subsystem library representations will consist of lower level models. Module design support will be the primary focus in definition of the design library elements. Backplanes, cables, etc. also will be included as library elements for fabrication documentation and, in some cases, for performance modeling.

The design library elements will be defined at the module and lower levels for implementing system upgrades. The lower level elements consist of logic function designs (unpackaged virtual implementations of processor kernels, memory arrays, interfaces, etc.), packaged logic functions (e.g., MCMs), electrical components (ICs, resistors), and mechanical components (thermal planes/cores, printed wiring boards, connectors). Library elements provided at these levels maximize hardware design re-use, ease the transition to new technology, and facilitate design re-use across different form-factors and packaging technologies.

Figure 6 illustrates examples of module, functional, and component level library elements. The most often used building block from the library will be the module level element. A module level library element will consist of the complete design, including the physical parameters required to fabricate, test, and integrate the module into a system. The module level library element models also include the functional and performance characteristics necessary for effective system design trade studies. Module designs will typically consist of lower level functions that also will exist in the library as modeled elements.

A functional element may serve as a virtual design that, together with other functional elements, makes up a module level design. Functional elements can include processor kernel designs and interface functions, as shown in Figure 6, or memory functions, support functions (such as test and maintenance control, analog-to-digital functions, timing and control, voltage regulation), or other functions as appropriate for use across multiple module implementations. Functional elements will not necessarily be fabricated as inventorable components, but the designs will be tested and verified for functionality as a part of the overall module design.

Functional elements may also be packaged as MCM or hybrid implementations. In this case, the functional design is also a physical implementation that can be fabricated and tested as a stand-alone component. Figure 6 illustrates an interface function that serves as a virtual function on the module and is also implemented as a MCM library component. The figure also shows that multiple functional elements may be implemented in a MCM package. In this example, the processor kernel design is implemented as a virtual functional element on a standard PWB and in a MCM. Other implementations could package a single processor kernel as a MCM component.

The examples in Figure 6 also can be used to illustrate the flexibility of library element approach in providing technology upgrades on model-year technology improvement cycles. The processor kernel architecture isolates the core processor (a TMS320C30 DSP in the actual implementation) from the other functions (sensor interface, external bus interface, memory). An upgrade to a TMS320C40 DSP as the core processor

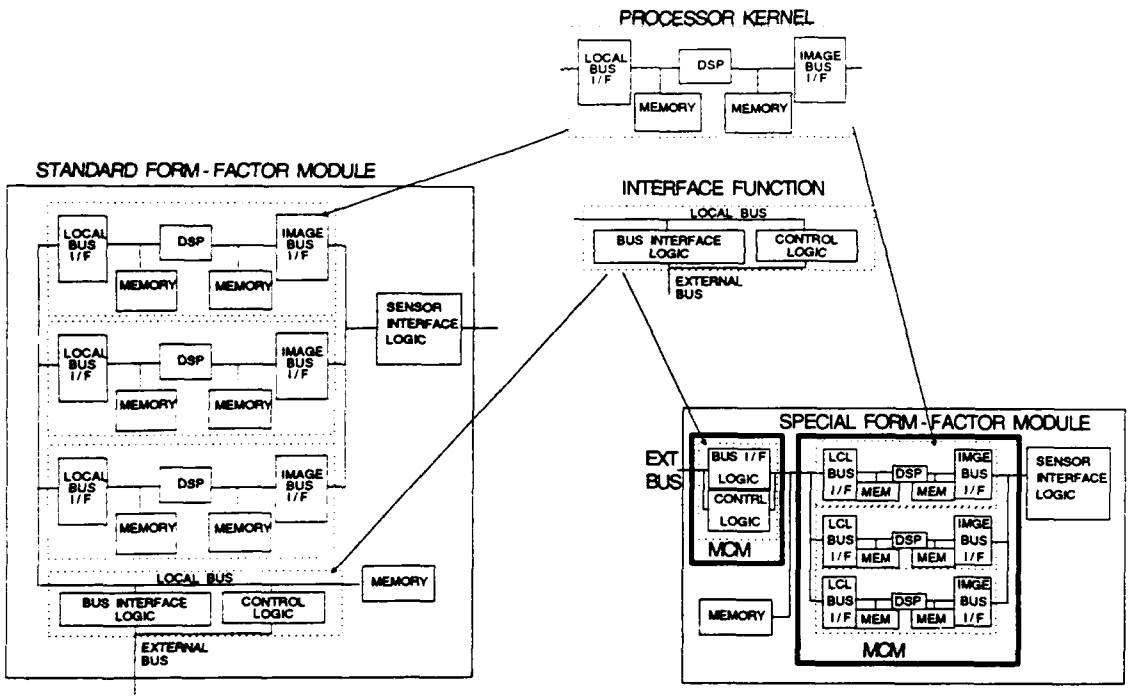


Figure 6
Library Element Model Examples

has no impact outside the processor kernel function because the processor kernel is designed as a functional element with a standard interface. Even changing the core processor to a non-TMS320 family component can usually be accommodated without affecting the other functions on the module. As another example, the interface function (a VME external bus in the actual implementation) can be changed to a different external bus (PiBus, Non-standard bus, Data Distribution Network) usually without impacting the processor kernel. The limitations in upgrading at the kernel level without impacting other functions are driven by the significance of changes in size, power, and functionality resulting from the upgraded kernel design.

Creating a library of designs at these levels provides rapid module design and upgrade cycle times, facilitates the transition of designs across multiple packaging form-factors and technologies, and allows extensive design re-use for module upgrades and new implementations.

2.3.5 RASSP Library Views

The RASSP library (repository) contains all the data of various types required throughout the life-cycle of RASSP products. As shown in Figure 7, these data have many views as defined by the requirements of the engineering, manufacturing, and test design activities, as well as library management.

The design view provides all the data necessary for the design process. The data contains models of the library components in formats suitable for:

- Design verification
- Placement and routing
- Performance modeling

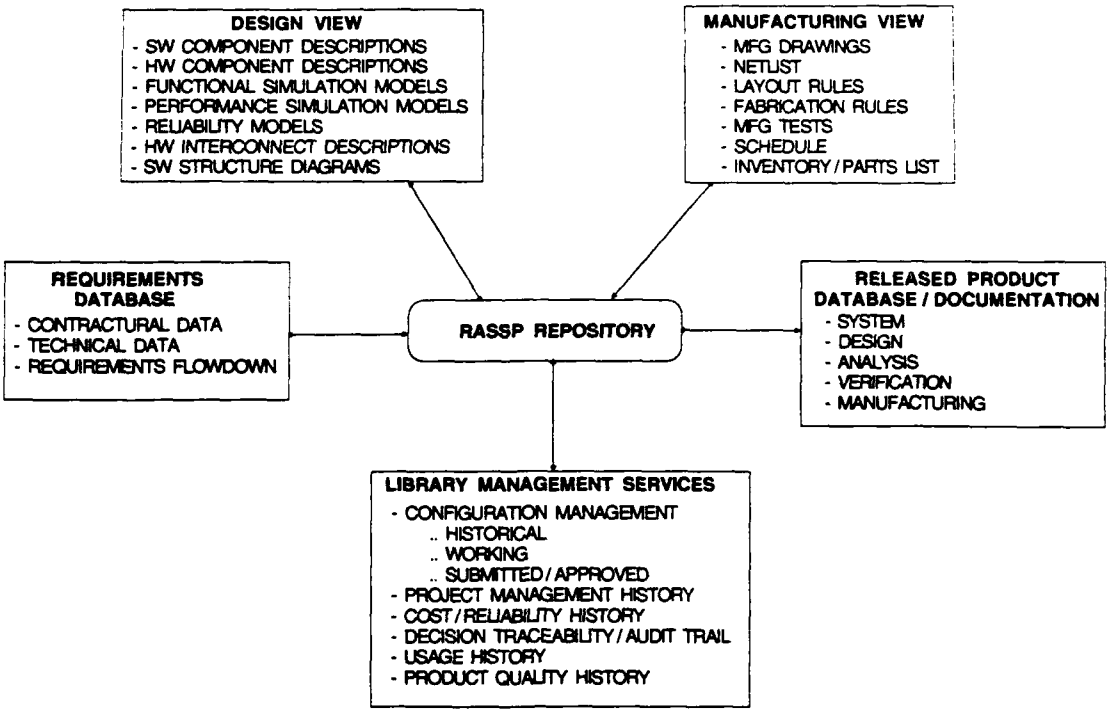


Figure 7
Library Element Model Examples

- Functional simulation
- Electrical analysis
- Physical analysis
- Reliability analysis
- Domain specific hardware and software construction
- All other aspects of design.

The manufacturing view provides all the data necessary to manufacture the processor. The data includes:

- Manufacturing drawings
- Interconnect netlists
- Manufacturing test patterns and test procedures
- Integration procedures
- Process flows.

The product release database contains all documentation associated with product design, manufacturing, or use. The data can include:

- Release drawings
- Test results
- Maintenance documentation and procedures
- User documentation.

The customer's data package is created from this data.

Library management includes the data necessary to maintain the library. This view provides data relevant to all out-of-cycle design activities, as well as library improvements and engineering changes.

The requirements data base provides requirements traceability for all the library components as they are designed. It also captures the requirements for the processor product that is in design.

Although the views in Figure 7 are shown as non-overlapping, in reality they will overlap extensively. For example, much of the manufacturing data used by the factory are identical to the design data created and used by the designers. The library ultimately must present all views and all data at the various stages of development to allow maximum flexibility and concurrency in the design and manufacturing process. The library in the diagram consist of federated, distributed design data. In the initial implementation phases of RASSP, much of the library data and library views will be maintained using currently available tools and databases.

2.4 Design Library Based Model Year Upgrades

An informal library based system development/upgrade capability has been useful in developing a family of signal processor modules and systems for several TI internal and DoD programs. Even with library information usually residing informally in people's heads or in standard (paper) documentation, the functional design element approach has been extremely effective in allowing design re-use and design for upgradability. With a formal repository for library elements, integrated design information, and better design tools to use the available information, the library approach will be very effective in reducing system development and upgrade cycle times.

2.4.1 Functional Element Design Examples

Figure 8 illustrates some of TI's experiences in developing and upgrading systems using an informal design library element approach. *Using designs based on design* library functional elements, the implementations have evolved through several upgrades involving a variety of scalar processors, TMS320 family DSPs, interfaces, module form factors, and software.

IR&D Demonstration System

The initial system implementation was an IR&D demonstration system integrating two dual TMS320C30 SEM-E form factor modules, data input and display output modules, and TI ATF 1750 and interface modules in an ATF Mission Display Processor (MDP) chassis, and demonstrating execution of Imaging InfraRed (IIR) target detection algorithms. Because we had access to design information during the development of the TMS320C30 (C30) DSP, we were able to complete the design and fabrication of an extended SEM-E dual-C30 breadboard module prior to availability of the first C30 devices. The final system demonstration used SEM-E form factor modules. With the excellent design information availability and good software development support tools early, the breadboard design was operational within a matter of days and the design released for SEM-E module fabrication. The total cycle from start of breadboard module checkout to fabrication of SEM-E form factor modules, integration into an MDP chassis, and algorithm demonstration, was less than six months. This is an example of an out-of-cycle development effort that allowed us to accomplish a rapid cycle prototype system upgrade to a Lockheed YF22 MDP chassis.

Tri-C30 Signal Processor

Using the dual-C30 module design experience, we developed a standardized C30 processor kernel architecture (essentially a virtual library element). We used this kernel design to implement a Tri-C30/1750 MCM

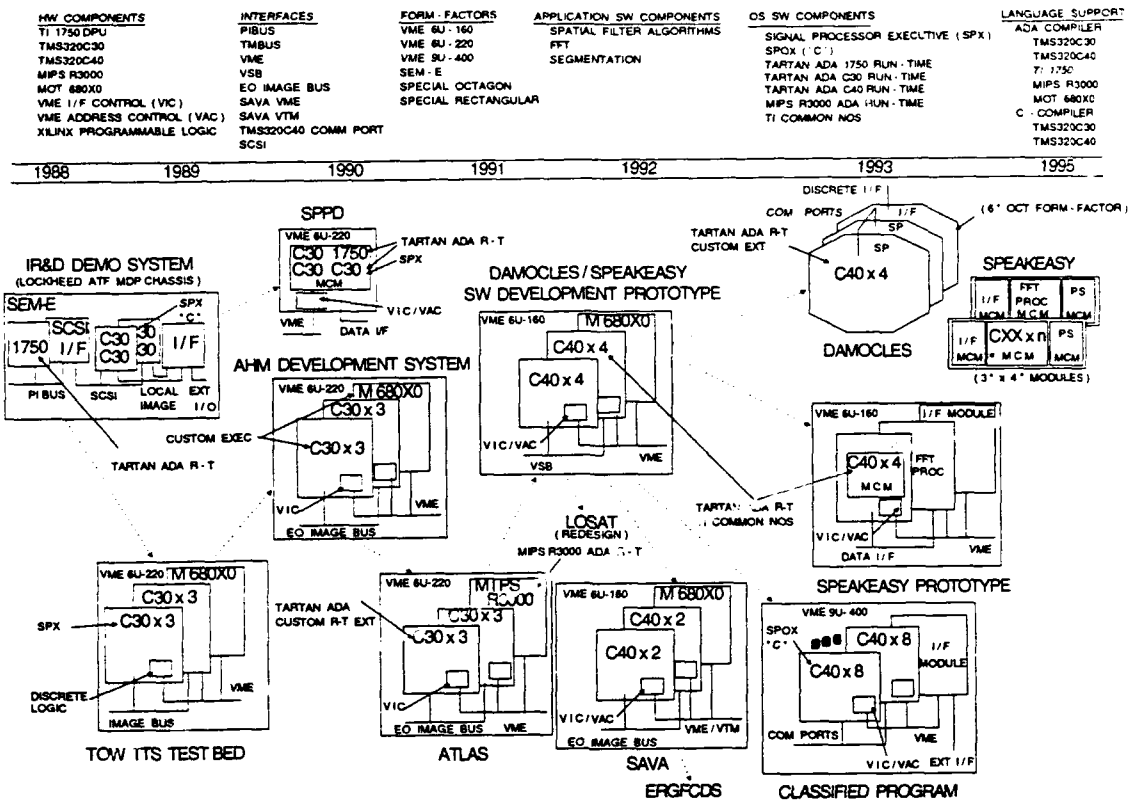


Figure 8
Functional Element Design Examples

packaged signal processor for the Signal Processor Packaging Design (SPPD) program, and a Tri-C30 VME form factor signal processor module for the TOW Tank Thermal Sight (TTS) testbed. The SPPD module was a stand-alone deliverable module, while the Tri-C30 VME module was integrated into a multi-processor system with a Motorola 68030 (M68030) scalar processor for data and control processing. The Tri-C30 VME module was then upgraded to enhance its VME bus interface. The original interface design used discrete logic and had limited functionality. The upgrade replaced the discrete logic with a VME Interface Control (VIC) chip (a new technology development) for greater functionality and the resulting module was used in the Anti-Helicopter Mine (AHM) development system. An additional system upgrade replaced the M68030 scalar processor with a Mips R3000 scalar processor module for use on the ATLAS program.

As hardware upgrades were implemented, the OS software also experienced several upgrades. The original Tri-C30 module used a Signal Processor Executive (SPX) that was a modified Array Processor Executive (APX) developed for the TI VHSIC array processor module, and was programmed in "C". The ATLAS signal processor module used the TARTAN C30 Ada run-time with custom run-time extensions for messaging and the hardware interfaces.

The transitions to new OS software and signal processor communications to different scalar processors was considerably eased by the use of interface standards and functional design elements. Defining a processor kernel function with standard local interface definitions isolated much of the processor design from the affects of changes in the hardware interfaces. The use of a common messaging interface design for

inter-processor communications isolated the software impacts of interface hardware changes to the hardware driver software. The remainder of the OS and application software was unaffected by the interface upgrades.

TMS320C40 Kernel Functional Element Designs

Another out-of-cycle design effort upgraded the Tri-C30 module design to a TMS320C40 based implementation. The C30 based standard processor kernel design was modified to take advantage of the technology enhancements in the C40 design (additional C40 Comm ports, enhanced addressing capability, faster memory ports). With advanced design information, fabrication of the initial module was completed prior to availability of the first C40 chips, and initial checkout was completed within days of receipt of the C40s. With this approach, we were able to implement a C40 based upgrade for the Tri-C30 module that shrank the form factor from a 6U-220 VME size to a 6U-160 VME size, almost doubled the computational throughput, further enhanced the VME bus interface, and changed the sensor data interface from an internally supported standard bus to an industry standard VSB interface. Successive implementations have employed the same basic processor kernel design with dual and octal processor configurations in different form-factors, packaging technologies, and with different data interfaces, operating system software, and using different programming languages.

Use of a standard design kernel with standard interfaces allowed rapid prototyping upgrades to the basic system architecture for this family of processors, even with an informal and manually intensive library based design approach. The capability to design using a formalized library of design elements with the tools necessary to port the designs to different form-factors and packaging technologies would provide a significant improvement in the hardware upgrade development cycle. Standardization of the software interfaces and an easily retrievable set of software library elements are also critical in significantly improving the software upgrade cycle time.

2.4.2 Standard MCM Library Element Examples

With higher density devices and packaging projected for the future, MCM level standardization is a viable approach for implementing a signal processor component library. Standardization at the multichip module level can be approached in several ways. Figure 9 shows a total processor on an MCM approach, and a building block MCM approach.

The Aladdin program produced a high performance signal processor MCM packaged in a round housing for missile applications. The URDA program will transition the Aladdin MCM to a surface mount approach and implement a dual Aladdin signal processor, double-wide SEM-E module assembly. Further technology upgrades will reduce the size of the Aladdin MCM and implement the interface circuitry in an interface MCM, allowing fabrication of a single-width SEM-E dual Aladdin signal processor module.

An alternative approach has been proposed for a future avionics application. Standard MCM functions are defined that, together, implement a signal processor module. Each MCM type can be implemented using core components selected from the library elements. The inter-MCM interfaces are standardized, which allows a mix and match approach to module implementation. All modules contain a data processor/interface MCM, global memory, on-module power regulator, and one or more signal processor MCMs. However, different global memory and signal processing functions may be implemented using the library elements. The global memory functionality required for radar applications, which may use a

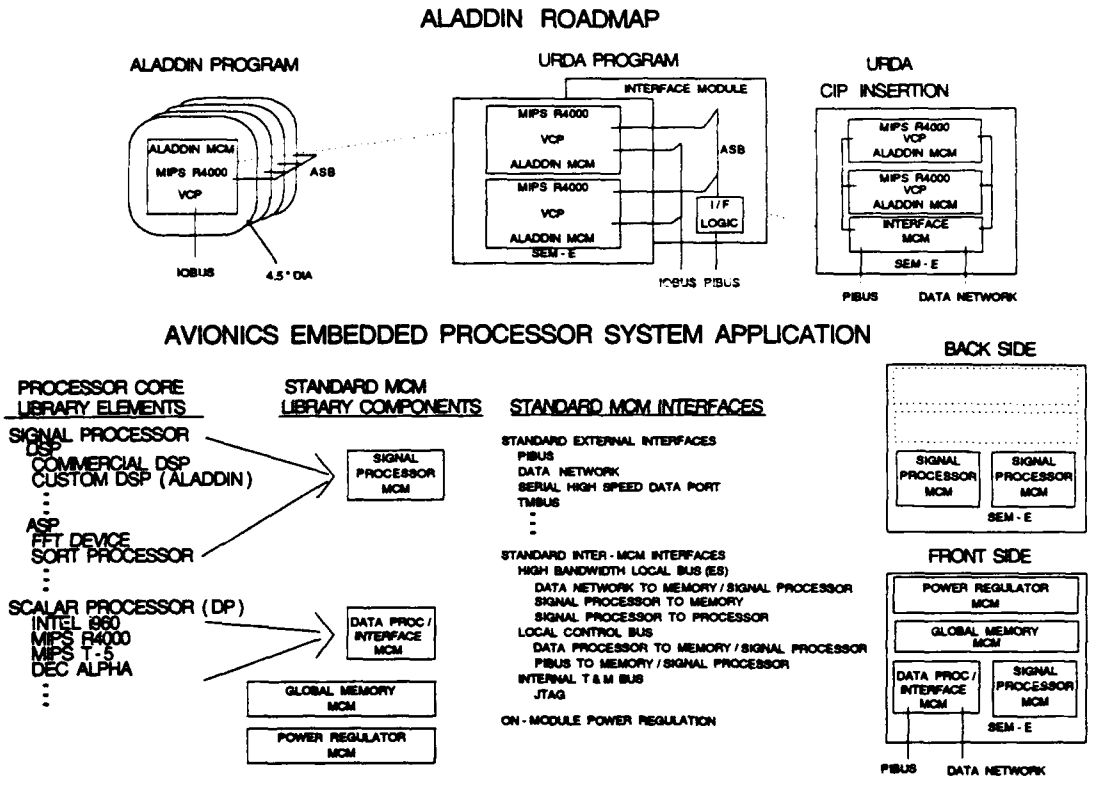


Figure 9
Standard MCM Library Element Examples

high bandwidth corner-turn memory, is different from the functionality required for image processing applications, which may use a large 2-dimensional addressed memory. Signal processor MCM functionality may range from algorithm specific processors, such as high performance FFT or sort enhanced processors, to highly programmable digital signal processors. Signal processor types may be mixed on a single module to meet the specific performance requirements of the application.

Both of these approaches employ library element designs and interface standards. Standard software interfaces and operating system designs are also included in these approaches. The result is an upgradable system design, allowing the best technology to be used in the functional implementations, within the constraints of standard interfaces.

2.5 Summary of Issues and Recommendations

Table 1 summarizes the primary issues in a RASSP implementation of a library based system design approach. Hardware and software interface standards are the main issues that need to be resolved in the RASSP implementation program. Actual development of library components is an ongoing effort, with an initial set of library elements developed under the RASSP implementation program.

Issues/Recommendations

Most of the hardware standard issues relate to selecting an appropriate initial set of hardware interfaces from the current industry and DoD supported interfaces. The selection of an inter-processor communication interface will most likely be from such candidates as the PiBus, VME,

Table 1 Issues/Recommendations Summary

ISSUES	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
HW Interface Standards Local buses (inter-module)	Limit number of local bus types, but no single standard	Yes	Yes	JIAWG IEEE VME64	ONGOING	Select standards from interfaces currently available, for demo program
Test I/Fs (component level & module/external level)	Standardize on one component level I/F (JTAG), minimize module level & external I/F standards	Yes	Yes	JIAWG IEEE SAVA		
External I/Fs (Sensor I/Fs, Comm I/Fs)	Limited number of external I/F types, but no single standard. Comprehend wire, fiber optic, serial, and parallel.	Yes	Partial	JIAWG (HSDB), no standardization on high speed (sensor data) I/Fs		
SW Interface Standards Application/OS	Standardize interfaces between the OS and the application SW	Yes	Yes	POSIX	1994-1997	Use results of POSIX, augment for the demo as necessary.
OS Core/HW Drivers	Standardize interfaces between the OS core functions and hardware driver SW	Yes	No			Define standards for the RASSP demo
Library Development	Define and develop functional design (HW & SW) library elements for design reuse. Develop module level library designs for use in RASSP upgrades.	Yes	No			Develop initial library elements as part of the RASSP program. Include HW (Module & Functional) and SW (OS & Application) in library.

Futurebus+, or possibly the F-22 Common Integrated Processor (CIP) Data Network Bus. For test and maintenance, the JTAG bus is the current industry wide standard at the chip level. The module interface will be either a TMBus or VTMBus. External communication interfaces can be selected from 1553, HSDB,or other interfaces currently in use on DoD systems. The external sensor data interface and the local signal processor sensor data interface, however, have very little in the way of interfaces to choose from that have widespread support. The CIP data network is applicable for the local sensor data interface, but there may be other candidates identified during the RASSP implementation phase. External interfaces may be selected from wire or fiber optic implementations, but again, there are few interfaces that could be considered widely supported standards at this time.

Operating systems interface standards are being addressed by the IEEE sponsored Portable OS Interface (POSIX) and ACM sponsored Ada Runtime Environment Work Group (ARTEWG) Catalogue of Interface Features and Options (CIFO). POSIX was originally intended for the non-real-time commercial market using a C interface. POSIX is now defining its interfaces in a language independent manner, with language bindings (e.g., C and Ada) added as desired. Due to the Next Generations Computer Resources (NGCR) basing its operating interface standard on POSIX, there is an attempt to add real-time interfaces within POSIX. There has been excellent progress in P1003.4, a, and b dealing with uniprocessor (and some multiprocessor with a shared memory model) real-time interfaces. Even though there is on going work in security and distributed communication, it is not clear that a suitable consensus for real-time military systems will occur in the near term. Performance demonstrations are necessary to show which POSIX interfaces are

applicable to real-time systems and to determine any required modifications and extensions.

The ARTEWG has also defined real-time interface extensions to the Ada runtime in its CIFO. These interfaces are optimized for Ada. These interfaces deal mostly in the uniprocessor environment. From an Ada perspective, the ARTEWG CIFO is a better choice. However, since the ARTEWG CIFO is limited to Ada it may not be as commercially acceptable as POSIX.

The RASSP implementation program should benefit from the POSIX and ARTEWG CIFO efforts. However, there will be OS interface areas not covered in the near future and care must be taken in applying the POSIX standards in their entirety until applicability to real-time is proven. There is little work addressing the OS to hardware interface standards. The JIAWG Input/Output Built-In-Test Description Specification (IOBIDS) defines a processor to PI-Bus interface. Additional standardization efforts in this arena would also benefit RASSP.

An initial library development effort will be required in the RASSP implementation program. Definition of the initial library elements will be dependent on the selection of the demonstration application and the characteristics of its application domain (typical algorithm computational requirements, processing and data flow, physical configuration constraints/requirements), and the extent of advanced technology available for the out-of-cycle design effort.

Summary

Development of a library based RASSP model-year upgrade approach is technically feasible. The selection of standards needs to be limited primarily to interface standards. Library elements (hardware and software) need to be specifically designed for re-use, and provided at the functional design element level. Following these basic rules, the design library approach will allow hardware and software design re-use and upgrades across different physical configurations and packaging technologies, system architectural evolutions, functional and performance upgrades, and different system applications. The real challenge will be establishing business incentives to maintain a flow of out-of-cycle library element updates consistent with the technology improvement cycle times.

3. DATA BASE AND DATA INTERCHANGE

This section focuses on the generic software architecture and software infrastructure of a RASSP system (not on the content of domain-specific representations or tools covered in other sections). This includes the framework and data base architectures needed to create a "seamless" RASSP development environment. Section 3.1 provides an overview of the RASSP software System Architecture. Section 3.2 is a summary of related work. Section 3.3 describes issues and recommendations specific to the RASSP software infrastructure. Section 3.4 summarizes the most important recommendations.

3.1 RASSP System Architecture Overview

Figure 10 shows a high level overview of the RASSP system architecture.

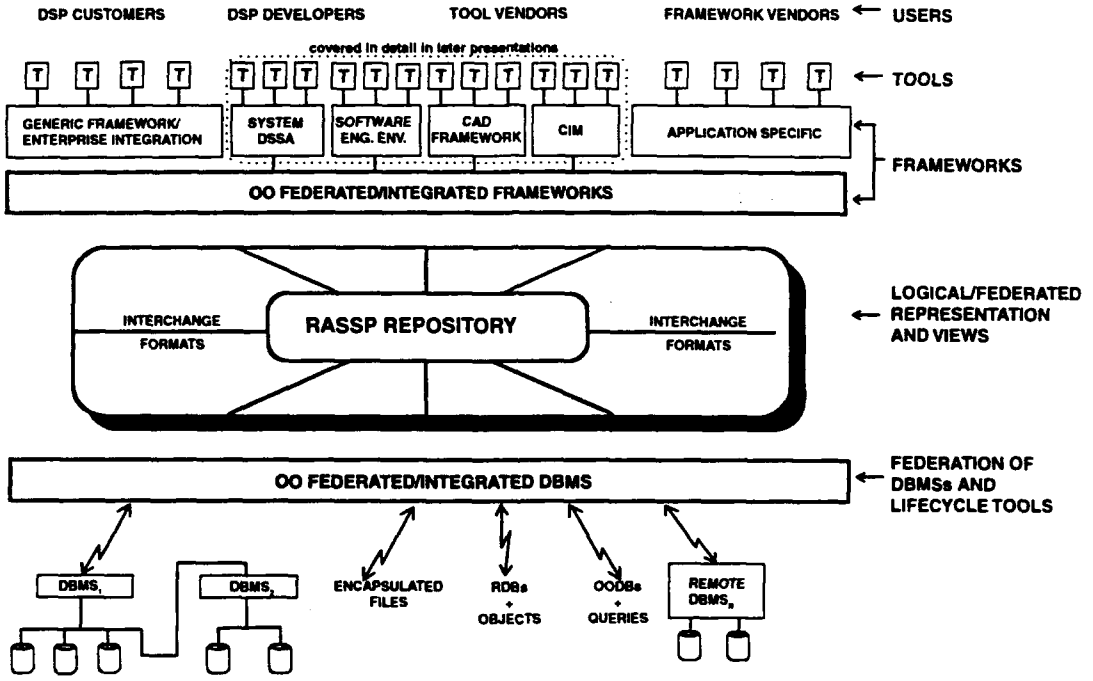


Figure 10
RASSP System Architecture

The previous section on design libraries made the point that central to the RASSP program is the development of RASSP domain-specific SW/HW libraries and standard representations, including interchange formats and object libraries. This logical data representation is shown in the oval in Figure 10. Much of the work in RASSP will be to develop or improve these domain-specific representations, working with other teams in a proposed RASSP consortium/working group and with broader communities like PDES/STEP and CAD Framework Initiative. We do not expect one uniform stable representation to result since improvements in tools will continue to require change. So the representation will be integrated where possible and federated elsewhere.

In this section we focus on the generic software architecture of RASSP systems. This includes the framework and data base architectures needed to create a "seamless" RASSP development environment.

In Figure 10, it is assumed that RASSP systems have several kinds of users. These include DSP customers, DSP developers, tool vendors, and various kinds of framework vendors, and data base vendors. One organization may simultaneously play several roles. Users are assumed to be geographically and organizationally dispersed. Some form of "trader" service will be responsible for sharing libraries and services, providing security and payment.

The tools and frameworks layer of the architecture is shown, modeled after the Object Management Group Architecture, as a "toaster model" consisting of a software backplane and a collection of plug-and-play tools and services. In Figure 10, generic and enterprise services are shown in the left-most framework; domain-specific system, software, CAD, and CIM services are shown in the middle (covered in later sections of this report); and capabilities specific to an application are shown on the right. The frameworks are shown as federated together. Services will be increasingly object-oriented. Industry consortia as well as CAx frameworks vendors are providing open or proprietary frameworks. Although integrated frameworks are the goal, in five years RASSP will still depend on federated frameworks where integration is not fully realized.

The data base layer is also shown as federated. Many RASSP tools and frameworks will use OODBs as OODB standards are put in place. RDB-OODB hybrids will provide the query capability needed for repository queries and for storing business views of RASSP systems. The RASSP "data base" (and repository) will be distributed and decentralized. New hybrid object-file systems may become important in providing integrated views of storage repositories that unify file systems, relational data bases, and object data bases.

3.2 Summary of Related Work

The RASSP program will not happen in a vacuum. It will depend on related work in several areas. As far as the software infrastructure goes, RASSP can provide a driver for demonstrating the integration of several generic technologies, but should reuse the latest solutions and work proactively to influence key communities that RASSP success will depend on; for instance, by identifying RASSP requirements and then influencing other DoD programs directed at these infrastructure technologies. RASSP should avoid significant funding of work targeted directly at proprietary infrastructure technologies.

In Section 3.2, we provide details on many of the SW infrastructure organizations RASSP needs to be aware of. Figure 11 provides a view of many of the infrastructure organizations and some CAx-specific organizations. We identify more CAx-specific groups in later sections of this report.

The RASSP consortium/working group needs to *participate* in core RASSP-related consortia and standards groups, including CAD Framework Initiative, PDES/STEP, and DARPA MADE.

The RASSP consortium needs to *monitor and influence* work in several areas, but much broader CAx (e.g., CAD, CASE, CIM, CAE) and generic needs will provide the market push to develop these technologies:

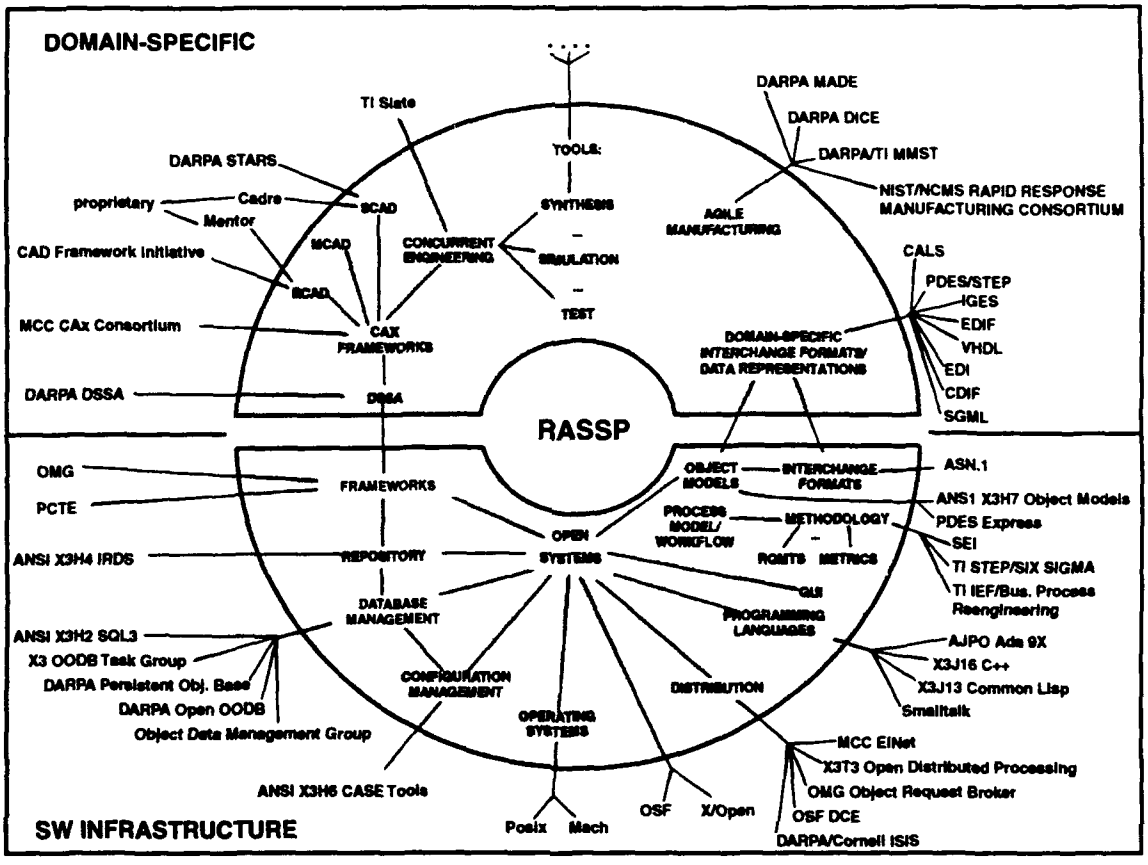


Figure 11
Summary of Related Work

- Generic frameworks like OMG and PCTE, which will eventually provide generic services that organizations like CFI and PDES/STEP, should specialize
- OODBs, federated data base gateways, enterprise-wide repositories, and distributed systems (including those distributed over wide area networks) for moving RASSP data between customers in the form/view required by RASSP tools
- Enterprise integration technologies for brokering goods and services and providing remittance
- Design methodology developments for software and hardware including tools for requirements capture, specification, object analysis and design, testing, and metrics for continuous improvement
- Coordination technologies for managing team interactions.

This section provides details on several areas of software infrastructure that RASSP will most directly depend on. These include:

- Enterprise Integration Section 3.2.1
- Information Management Standards Section 3.2.2

- Generic Frameworks — Section 3.2.3
- Domain-Specific Frameworks — Section 3.2.4
- Data base Management Standards — Section 3.2.5
- Object-Oriented Data base Systems — Section 3.2.6
- Distribution — Section 3.2.7
- Generic Interchange Formats — Section 3.2.8

3.2.1 Enterprise Integration

The RASSP community will need to share libraries of reusable designs and specifications, and also share software services and tools developed remotely. Not all portions of a design may be available locally nor will all processing capabilities. Enterprise Integration is one key to competitiveness in the 1990's. This means putting in place "information highways and byways" that are navigable but secure and that cross organization boundaries (see Table 2).

Table 2 Enterprise Integration

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
CALS Initative	CALS standards • IGES, EDIF, VHDL PDES/STEP, SGML, ODA • IRDS, SQL, RDA, GOSIP, OSF NCS RPC, POSIX.1, X, EDI	• Alignment with CALs stands	On going	Doug Lawson 214-575-2682		• Goal: Electronic storage and interchange of product data between contractors and customers
MCC	ElNet Services	• Broker RASSP libraries and services	c1994	Bob Peterson 214-995-6080 peterson@cac.b.com	Bill Thompson (MCC) 512-338-3353	• Goal: National enterprise integration network to support building a global info systems marketplace • Built on TCP/IP internet comparable • Will support -WAN access -Authentication (via MIT Kerberos) -Secure mail -Directory services (brokering commercial services and products) -Remittance
Thinking Machines	Wide Area Info. Server (WAIS)	• Index RASSP libraries and services • Content-based navigation	In place	Bob Peterson 214-995-6080 peterson@cac.b.com	Public domain	• Indexes semi-structured files • 100's of info servers; 20* countries
ANSI X12	Electronic Data Interchange (EDI)	• Broker RASSP libraries and services	Now	Chem Musser 214-575-2182		• Interchange standard for inter-company business transactions (e.g., quotes, purchase orders,)

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
MCC	Camot Project	• Distributed info management	1994	Linda McCalla 214-995-0783	Bill Thompson (MCC) 512-338-3353 Phil Cannata	• Agents: work flow, legacy system access • Build on standards
ANSI X3T3	Open Distributed Processing (ODP)	• Broker/tracke r standards	c1995	Craig Thompson 214-995-0347	Ed Stull, Chair 301-942-4355	• Standard for trading/brokering object services and products
CCITT	X.500 Directory Service	• Location-based navigation	Now	Drew Holland 214-575-2620		• Directory is hierarchical organized geographically and organizationally • Extensible

One force for enterprise integration is the CALS initiative. The CALS initiative objective is electronic storage and exchange of data among DoD and its government contractors. CALS is an umbrella of interchange standards (e.g., IGES, EDIF, VHDL, EDI, and SGML and ODA for documents) and application program interface (API) standards (e.g., IRDS, SQL, SQL RDA, GOSIP, RPC, POSIX, and X).

Another important vision of enterprise integration comes from the industry-led report "21st Century Manufacturing Enterprise Strategy," Volumes 1 and 2, partially funded by the Office of the Secretary of Defense Manufacturing Technology (MANTECH) Program and published in November 1991. This report describes "agile manufacturing" as a key to competitiveness and "virtual companies" as a model for competing by relying on data and operations available remotely in other organizations.

Several sorts of infrastructure software are becoming available to make these visions possible.

MCC EINet Services and ANSI X3T3 Open Distributed Processing are focusing on putting in place a global "trader" infrastructure so products and services can be exchanged electronically, providing distribution, security, and payment. For example, one EINet project being worked on by ATLAS, an MCC subsidiary, is an electronic databook.

X.500 provides extensible standards for namespaces that address geographically and organizationally remote objects via navigation. The public domain Wide Area Information Server (WAIS) from Thinking Machines provides a complementary way to address remote information by content by indexing semi-structured files. There are hundreds of WAIS servers in more than twenty countries. WAIS could be used to index RASSP libraries for content-based access. X.500 could be used to name RASSP objects remotely.

ANSI X12 Electronic Data Interchange (EDI) provides standard forms for quotes, purchase orders, and buying and selling products electronically. This forms the basis for an electronic marketplace and may affect procurement and acquisition strategies.

MCC Carnot project is one of many working on several levels of enterprise integration, including federating heterogeneous data base management systems and process description languages.

3.2.2 Information Management Standards

Standards groups are active in software infrastructure areas generically related to RASSP software System Architecture (Table 3).

Table 3 Information Management Standards

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
ANSI X3H4	IRDS (repository)	Repository Standards	On going	Bob Hodges 214-575-3442 hodges@lobby.ti.com	Jerry Winkler winkler@nasa mail.nasa.gov	• Object services architecture • Mappings between data models • Representation foundations
ANSI X3H6	CASE Tool Integration Models	Configuration Management	Now-1995	Doug Conley 214-5754578	Cathy Chapman (Dec) 603-881-1705	• 00 change management for large grain objects
ANSI X3H7	Object Model Interoperation	Object Model Standards	On going	Craig Thompson 214-995-0347 thompson@cac.ti.com	Craig Thompson 214-995-0347 thompson@cac.ti.com	• Reference model for object model harmonization • Architectures for mapping between object models

- ANSI X3H4 -- Information Resource Dictionary Systems (IRDS). This groups focuses on repository standards, including object service architectures, mappings between data models, and representation foundations (partly in conjunction with the DARPA Knowledge Representation Standards Initiative). Texas Instruments is active on this committee.

- ANSI X3H6 -- Case Tool Integration Models, formerly the industry consortium CASE Integration Services (CIS). This group focuses on configuration management and CASE tool services. Texas Instruments is active on this committee.

- ANSI X3H7 -- Object Information Management. This new group focuses on object model interoperation issues and harmonization of object models. Texas Instruments is active on this committee. See Section 3.3.1.

There is a fair overlap in the work of groups listed in Sections 3.2.2, 3.2.3, and 3.2.4. See Section 3.3.2.

3.2.3 Generic Frameworks

Refer to Table 4.

Table 4 Generic Frameworks

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
Object Management Group (OMG)	• IDL/object model • CORBA Object Request Broker • Object Services Architecture	Open (non-proprietary) framework	CORBA 92 Object services 1993-1994	Craig Thompson 214-995-0347 thompson@csc.ti.com	Richard Soley (OMG) 508-820-4300 soley@omg.org	• Wide spread industry support • 225 companies including major platform and database vendors • Generic framework
ECMA TC33	• Portable Common Tools Environment (PCTE)	Open (non-proprietary) framework	Now		Ian Thomas iasman@sda.com	• In use in European Community • Entity-Relationship data model • Coarse-grain objects (files)

Industry consortia (and standards groups) are working on application integration frameworks. The idea of a framework is to provide an environment where common services are available to build applications (e.g., common data base, user interface, help system, etc.) making it much easier to build look-and-feel compatible applications.

Object Management Group (OMG), founded in 1989, is an industry consortium with around 250 members, working on an object-based framework. Major players like HP and Sun are active in OMG. Texas Instruments has been active in OMG since early 1990. We have influenced OMG's overall Object Management Architecture Guide, are editor of the OMG Object Services Architecture, and contributed to the OMG Object Model. Other important OMG work includes the OMG Common Object Request Broker (CORBA), which provides an OO distribution framework for sending messages between distributed applications.

Portable Common Tools Environment (PCTE) is an older, more mature framework developed in Europe but gaining popularity in the U.S. While current PCTE uses an entity-relationship model and is coarse-grained (file-based), a newer effort called PCTE+ aims to make PCTE object-based.

ANSI X3T5 Open Systems Interconnection (OSI) is also defining a collection of services for systems management and "managed objects."

So far, these three groups do not have strong liaisons with each other and have not established clear road maps of how to build on each other's work. See Section 3.3.2.

3.2.4 Domain-Specific Frameworks

One of the lessons learned in the 1980's is that problems we cannot solve generically can often be solved in specialized domains. Table 5 lists several important efforts to put application-specific frameworks in place for electrical, mechanical, software and other application areas.

Table 5 Domain Specific Frameworks

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
DARPA	Domain Specific Software Architectures (DSSA)	Domain-specific frameworks	1991-1997	Doug DeGroot 214-575-3763	Erik Mettala (DARPA) 703-696-2219 mettala@darpa.mil	• Domain analysis • Domain-specific objects and operations
MCC	CAx Consortium	Open CAx framework standards	Now-1995	Mark Esteve 214-696-1217		
CAD Framework Initiative (CFI)		eCAD framework standards	On going	Mike Mahoney 512-250-6845		• eCAD standards for framework and interchange formats
MCC ATLAS Standards Lab	Independent subsidiary of MMC	RASSP infrastructure	On going		Bill Thompson (MCC) 512-338-3353	• Objective: accelerate and coordinate open enterprise information integration standards • Manages CAD Framework Initiative Pilot Program including CFI/ATLAS Electronic Data Books project
Engineering Info. System (EIS)	EIS	Early CAx system	1985-1991	Craig Thompson 214-995-0347 thompson@cac.tl.com		• USAF contract - Honeywell/Xerox • Precursor to CFI, OMG,
PDES/STEP	• Architecture • EXPRESS object model • Interchange formats	Product data interchange standards	On going	Neal Stollon 214-575-2819	Pete Brown (NIST)	• Key group to develop eCAD/mCAD/ CALS standards for product data interchange
Mentor	Falcon Framework	CAD framework plus tools	On going	Neal Stollon 214-575-2819	John Schwartz (Mentor)502-426-7000	• Proprietary CAD framework • C++ class library
DEC	Powerframe	CAD framework plus tools	On going	Neal Stollon 214-575-2819		
DARPA STARS	Repository	SW reuse repository	On going	Angela Harper 214-575-6205		• Ada Repository

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
CADRE		IPSE/CASE framework	On going	Angela Harper 214-575-6205		• Proprietary IPSE/CASE
Electronic Industries Association (EIA)	CASE Data Interchange Format (CDIF)	CASE Interchange Format	Now, On going	Doug Conley 214-575-4578	Mike Imber, Chair	• CASE interchange format
TI	STEP	SW Quality Improvement	On going	Marie Silverthorne 214-995-0346	—	• Works closely with SEI (maturity model metrics)
DARPA	DICE/MADE	Leverage DARPA technology	On going	Bob Peterson 214-995-6080	Mike McGrath 703-696-2224	• Concurrent engineering research • Brokering infrastructure
NIST/NCMS	Rapid Response Manufacturing Consortium	Leverage mCAD technology	1992-1995	John Richardson 214-958-6513		• Feature-based modeling tools
DARPA	MMST	Factory modeling	1992	John McGehee 214-917-2189		• Modular front end

The DARPA Domain-Specific Software Architectures program can be viewed as an effort to develop domain analysis and domain-specific application generators for reuse across important application domains (e.g., missile or tank development). In this sense, the RASSP program is exactly a DSSA program in the signal processing area.

Of course, seen from this light, RASSP should benefit from longer standing DSSA-like programs in the CAD, CASE, and CIM areas. Framework architectures in each of these environments already exist and RASSP needs to align with these systems. This is a complex and confusing area. There are several consortia as well as more mature proprietary frameworks to choose from. The newly-formed MCC CAx Consortium was formed in recognition that many of these efforts contain common generic framework elements. Also see Sections 3.2.2 and 3.2.3.

PDES/STEP is working on a number of these standards, targeting those that describe "product data" from several views including graphical, electrical, and mechanical, focusing on developing data interchange formats. As a specification language, they have developed the object-oriented Express language, and an Express-to-STEP Data Access Interface (SDAI)/C++ binding is in progress. The RASSP effort should be aligned with this effort and should proactively work on RASSP-domain generic representation standards through participation in the PDES/STEP effort.

The PDES/STEP Express language is finding some acceptance in other areas as well. The Petrotechnical Open Software Corporation (POSC) is a consortium of leading oil companies and vendors seeking to develop defacto standards for industry specific computing. They are planning to develop industry-specific data models using Express. These data models will be used to create data base schemas for object data base systems. This standardization of industry models is expected to provide a platform for reducing the cost of developing new applications, repositories and frameworks. This type of modeling should be valuable for other industries (e.g., aerospace), and provides a blueprint of how U.S. industries could build industry-specific representations to allow information interchange.

There are also important proprietary software frameworks, representative important ones being Mentor Falcon for eCAD (electronic design) and Cadre Teamwork for sCAD (or software CAD, or IPSE/CASE) (see Section 3.3.2). These frameworks are typically closed, proprietary architectures (though vendors may sell "openness" for a price) but they tend to be better populated with domain-specific tools.

DARPA has other programs that also need to influence these CAx-specific frameworks efforts, including DARPA DICE and DARPA MADE as well as specific programs like DARPA MMST at Texas Instruments. Another related program is the NIST/NCMS Rapid Response Manufacturing Consortium, including Ford, GM, Texas Instruments, and several small mCAD and manufacturing companies and organizations, formed in 1992 to improved feature-based modeling tools in the mechanical CAD domain. Finally, the DARPA STARS program is building an Ada-oriented software environment, a kind of software-specific framework.

Not all domain-specific framework architectures will succeed. Sematech embarked on an interoperability framework architecture but drew back when it realized that its leverage in the CAD industry would not allow enough momentum to develop yet-another-framework and that it should reuse existing efforts in this area. It is argued in Section 3.3.2 that CAD Framework Initiative and PDES/STEP should explicitly plan to reuse more horizontal frameworks and not re-invent their own but focus instead on domain-specific representations and tools to populate such frameworks. RASSP should do this as well. A RASSP working group should

frameworks. RASSP should do this as well. A RASSP working group should early on plan to identify key frameworks (we recommend OMG, CFI, and PDES) and focus on working through these groups to insure its more generic framework needs will be met.

3.2.5 Data Base Management Standards

Much of the data base management software infrastructure that RASSP will need is in place or moving into place. RASSP will depend on data base systems that can store and concurrently share design, life cycle, and other sorts of "objects". There is a requirement to associatively query at least repository data for reuse. In the next five years, hybrid OODB-RDBs should support the basic RASSP DBMS requirements in functionality and performance. The biggest current data base management hole, where more work is needed and not enough work is underway, is in the area of federated, decentralized, scalable, inter-enterprise data base systems. Issues related to this area are described in Section 3.3.5 and the state-of-the-art in this area is described in Section 3.2.5.3. The rest of this section surveys the state-of-the-art, trends, and relevant standards in data base systems.

3.2.5.1 Relational Data Base Management Systems. One problem with relational data base systems has been that they are not currently the DBMS of choice for many sorts of engineering data, which tend to use object representations either stored in traditional file or interchange format-based approaches or in newer object-oriented DBMSs.

To make SQL more attractive for these sorts of applications, the relational DBMS community is adding objects to its SQL standard. ANSI X3H2 SQL Technical Committee is responsible for standardizing SQL, and is currently developing SQL3, which is scheduled to become a standard in 1995 (see Table 6). This work is just beginning in 1992. Current proposals are to adopt an object model that is close to a subset of C++. RASSP may be able to use SQL3 to retrieve objects from reuse libraries, as long as the libraries are centrally stored. This raises the "yet-another-object-model" issue discussed in Section 3.3.1. The problem is, the RASSP object data (and CAE data in general) will need to be mapped into SQL3 objects to be operated on while in the library. This may cause problems if the RASSP object model(s) do not match the SQL3 object model. Another potential problem is performance: will RDB products, engineered for set-oriented operations, provide the same performance as object-oriented data base systems that are engineered for good performance on navigation operations?

3.2.5.2 Object-Oriented Data Base Management Systems. Object-oriented design methodologies and programming languages are quickly becoming mainstream. Object-oriented data base systems (OODBs) complement these by providing functions that allow applications to store, share, and navigate through collections of persistent objects.

A variety of small OODB vendors are now competing (see Table 7). Many provide more-or-less seamless access to C++, Smalltalk, Common Lisp, and/or Ada. "Seamlessness" means that the object model of the data base is the same as the object model of the programming language, making it much easier to use a DBMS, and dramatically reducing the cost of mapping program objects to data base objects. Some OODB vendors have their own proprietary object models (e.g., Itasca). Many are ground-up DBMS systems; some are OO interfaces to relational systems (e.g., HP OpenODB). Approaches to achieve persistence vary from memory mapping approaches (e.g., Object Design) that are fast but potentially may introduce safety problems inherited from host programming languages like C++, to translation-based schemes (e.g., Versant, Objectivity, Gemstone, etc.) that are more portable across heterogeneous environments.

Table 6 Data Base Management - Standards

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
ANSI X3H2	SQL1, 2, 3	Relational DBMS standards	SQL2: 92-94 SQL3: 95	Doug Conley 214-575-4578	Don Deutsch, Chair 301-340-4580	• SQL2 current RDB standard. NIST is specifying adoption schedule • SQL3 (w objects) c1995 • Oracle, DEC, NIST are prime movers
ANSI X3H2.1 (SQL Access Group)	Remote Data Access	DBMS federation	1993-1994	Doug Conley 214-575-4578	Don Deutsch 301-340-4580	• Standardizing dynamic SQL, views of catalog, exceptions, eventually distributed query • Competing with IBM DRDA
X/Open	XA	Transaction federation standards	1993-1994	Ron Sellars 214-575-4877		• Transaction coordinator standard for transactions spanning multiple DBMS systems
ANSI X3 OODB Task Group	OODB Reference Model	OODB standards	1991	Craig Thompson 214-995-0347	Craig Thompson, TI V-Chair 214-995-0347	• Characterizes OODBs
Object Data Management Group	OODB Standards Proposal	OODB	12/92	Craig Thompson 214-995-0347	Rick Cattell, Sun Chair 415-336-	• Common OODB API from consortium of OODB vendors • OMG IDL interfaces included

The years between 1985 and 1990 were formative years for OODBs. Several small vendors emerged with different kinds of OODB products but industry did not immediately commit to what appeared to be somewhat experimental systems. Problems were a lack of understanding of exactly what OODB systems were and a lack of OODB standards. These problem are now being addressed.

In 1991, the X3/SPARC/DBSSG OODB Task Group completed a two-year project to develop an OODB Reference Model and also a road map for OODB standardization. The purpose of the OODB Reference Model is to provide a clear characterization of the functionality of OODB systems. The purpose of the road map is to provide X3 with recommendations for where consensus that can lead to standards is possible and desirable in the OODB area. One of the direct outcomes of the OODB Task Group work was the formation of ANSI X3H7 Object Information Systems discussed in Sections 3.2.2 and 3.3.1. TI's DARPA Open OODB project provided a vice-chair for OODB Task Group and served as co-editor for the group's final report.

In 1992 a working group of OODB vendors came together to form the Object Data Management Group (ODMG) to develop a much-needed strawman application program interface (API) standard for OODBs. This work is currently progressing well but is "ODMG Confidential" with limited distribution. A public release is planned for December 1992. TI's DARPA Open OODB project is a reviewer.

OODB systems are now becoming more main stream, and OODB vendors are beginning to announce strategic partnerships. For instance, Versant is part of IBM's repository strategy; Informix is using HP OpenODB to provide an object interface to its RDB; NeXT has announced an agreement with Object Design; and Lucid's Energize C++ programming environment is also built on Object Design. Some CAD vendors, like Mentor (e.g., in the Mentor Falcon Framework), agree that commercial OODBs are the right direction but provide default proprietary OODBs that will probably be replaced with commercial OODBs depending on their customer's needs and market directions. By architecting a system to be OODB-independent and

Table 7 Data Base Management - OODB Vendors

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
Versant Object Technology	Versant ODBMS	OODB for C++, Smalltalk	Now	Craig Thompson 214-995-0347 thompson@csc.ti.com	Mary Loomis 415-329-7539 mloomis@csc.com	• Licensing agreement for IBM AIX CASE for PCTE • Versant-Sequent-PARC Place Joint Dev Agreement • Development environment • Gateway to Oracle • PDES/STEP, commercial MIS • OA&D tool - Rational's Rose (Booch)
Objectivity	Objectivity/DB	OODB for C++	Now	Craig Thompson 214-995-0347 thompson@csc.ti.com	Drew Wade 415-688-8000 drew@objy.com	• Focus on ECAD, CFI • Heterogeneous distributed platforms • No query language, no record level locking • Integrated DEC FUSE, part of Cohesion CASE tool • Corp. partners: Valid Logic,
Object Design	Object store	OODB for C++	Now	Craig Thompson 214-995-0347 thompson@csc.ti.com	Tom Atwood 617-270-9797 tom@ods.com	• Homogeneous nets only • License for IBM EDA tools • Embedded in Lucid Energize C++ prog. env. • Corp. partners: Mentor, NCR, Microsoft, Kodak • Parameterized types for C++ licensed to AT&T • NeXT OODB standard
Servio	Gemstone	OODB for Smalltalk, C++	Now	John McGehee 214-917-2189	Jacob Stein 510-814-6266 steinj@slc.com	• 4 GL/Geode graphical dev environment • IBM Business Partner (manufacturing) • MMST, case, telecom, aerospace • Agreement with Sybase, Neuron Data • Stores objects methods • Can share Smalltalk, C++ objects

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
Ontologic	Ontos	OODB for C++	Now	Craig Thompson 214-995-0347 thompson@csc.ti.com	Tim Andrews 617-272-7110 tim@ontos.com	• Homogeneous nets only • IBM Business Partner • Studio Development tool
GIP Altair	O2	OODB for C02	Now	Craig Thompson 214-995-0347 thompson@csc.ti.com	Francois Bancilhon 33-1-30-84-77-77 francois@02.tech.fr	• Multiclient, single service • Application development environment
Persistence Software	Persistence	Presistent C++	Now	Craig Thompson 214-995-0347 thompson@csc.ti.com		• C++ class library
UniSQL	UniSQL/X, /m	OODB/RDB hybrid	Now	Craig Thompson 214-995-0347 thompson@csc.ti.com	Won Kim unisql! kim@ca.utexas.edu	• Combine OO and RDB • UniSQL/4GE • UniSQL/M - multiple RDBs
HP	Open ODB (IRIS)	OODB I/F to RDB	Now	Craig Thompson 214-995-0347 thompson@csc.ti.com	Bill Kent 415-857-8723 kent@hplabs.hp.com	• Built on HP Allbase RDB • Sets required • Interface to other RDBs planned Informix • Proprietary object model
Itasca		OODB for Lisp	Now	Craig Thompson 214-995-0347 thompson@csc.ti.com	Doug Barry 612-851-3158 doug@itasca.com	• Developed at MCC • Supports Lisp • Proprietary object model
Software Productivity Solutions	Classic Ada with Persistent	Persistent OO Ada	Now	Craig Thompson 214-995-0347 thompson@csc.ti.com	Andy Rudmick	• Uses a preprocessor • Must interface to an OODB

also by adopting emerging OODB standards, systems like RASSP will be able to remain OODB-independent to a reasonable extent.

Table 8 provides a view of relevant R&D in the area of data base management. DARPA's research program in the data base area is the Persistent Object Base (POB) program initiated by Dr. Erik Mettala (DARPA/SISTO) and now under the direction of Dr. Gio Wiederhold (DARPA/SISTO). This program is funding coordinated work at Texas Instruments, University of Wisconsin, MIT, Brown, Oregon Graduate

Table 8 Data Base Management - R&D

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
DARPA	Persistent Object Base Program	OODB	On going	Craig Thompson 214-995-0347	Gio Wiederhold (DARPA) 703-696-2218 Erik Mettala (DARPA) 703-696-2219	• Database system R&D at TI MIT Brown, OGC Wisconsin Colorado Utah, CMU
DARPA/NIST	OODB Test bed	OODB Standardization	1991-1994	Craig Thompson 214-995-0347	Elizabeth For.g (NIST) 301-975-3250	• Functional eval suite for OODBs
DARPA/TI	Open OODB	Reconfigurable OODB/framework tool kit	4/93	Craig Thompson 214-995-0347	Craig Thompson (TI) 214-995-0347 Gio Wiederhold (DARPA) 703-696-2218	• Modular OODB architecture • Reconfigurable • Use/improve what you need • OODB/RDB hybrid • Distributed server

Center, University of Colorado, University of Utah, Carnegie Mellon University, and NIST.

The DARPA POB Open OODB project, being executed at Texas Instruments, is developing a modular OODB that bridges the gap from OODB to framework architectures. The system is structured as a collection of object services that can be configured as an OODB system. A beta release of the Open OODB will be available from Texas Instruments in April 1993. It is intended that the Open OODB form a testbed that allows users the ability to configure systems with only the modules they need and to improve or modify modules as required for increased functionality or higher performance. The DARPA/NIST POB Testbed will provide OODB functionality test suites. A PDES Express Testbed effort is planned to provide Persistent Express using the Open OODB.

3.2.5.3 Hybrid RDB-OODB Data Base Management Systems. OODB vendors are adding SQL query capabilities to be able to compete in the relational data base market. As mentioned, relational vendors are adding object data base capabilities.

The Petrotechnical Open Software Corporation (POSC) is a consortium of leading oil companies and vendors seeking to develop defacto standards for industry specific computing. In September 1992, they completed an evaluation of potential data bases for objects and selected UniSQL and HP's Open ODB. Both of these data bases combine relational technology and object technology. HP uses a layering approach while UniSQL provides integration at a deeper level.

The TI Open OODB project has developed an SQL-based Object Query Language module that can be ported to OODB systems from Versant, Objectivity, etc. to provide similar hybrid integration. It makes use of the DARPA EREQ Query Optimizer generator from the University of Colorado.

In the five year time frame, this trend toward convergence makes it a non-issue whether to adopt OODB or RDB technology. But convergence of OODB and RDB interface standards may still be an issue in five years.

3.2.5.4 Scaling up to Enterprise-wide Federated Data Base Management Systems. Some development and standards work is going on in the area of scaling up data base systems to provide enterprise-wide and even inter-enterprise data management solutions. The RASSP architecture shown in Figure 1 pictures a hybrid, federated data base management layer that

provides a uniform DBMS interface to RASSP applications. This is the weakest area in the data base management portfolio.

Progress is being made in some key standards areas (see Table 6).

ANSI X3H2.1 Remote Data Access (RDA), a corresponding industry group called SQL Access Group (SAG), and IBM Distributed RDA are all working on standards in support of distributing SQL-based relational DBMS systems. These include standardizing subsets of SQL, a standard "call level interface" for connecting to SQL data Bases, executing dynamic SQL commands, and accessing standard SQL system catalogs. Some PC-based products, like MicroSoft Access and Pioneer Q+E, are providing tools like Open Data Base Connection (ODBC) that implement the SAG call level standard to provide uniform access to multiple, possibly remote SQL data base systems.

X/Open is an industrial consortium that is putting in place a suite of de facto standards that guarantee a reasonable degree of product interoperation for products branded as X/Open conformant. X/Open is working on an industry standard transaction processing protocol, called XA, that defines an open two phase commit protocol. This will permit operations on complying heterogeneous DBMS systems to commit or abort correctly even though the scope of the transaction crosses DBMS boundaries. This is another important step in scaling up DBMS systems to support enterprise-wide operations.

Demonstrations of transactions containing a mix of relational and object-based data base operations are still needed. Also, much more work on open optimizers is needed to support distributed queries, where part of a query is executed in one DBMS and part in another, or where relational joins across DBMS boundaries are supported.

Some research work is underway in next-generation "object file systems" that combine the benefits of traditional file systems with the functionality of OODB systems. The idea is to provide a migration path from today's simple file systems to object systems that provide strong typing, behavioral extensions, queries, etc. Microsoft is working on an object-file system. Sun is working on a strategy called "Distributed Objects Everywhere". There is research progress at University of Wisconsin and some work at Texas Instruments. ECME PCTE provides an entity-relationship view of a file repository.

It is likely that much progress will be made in the next five to ten years in the area of federated enterprise-wide DBMSs but this is a fertile area for accelerating progress. There is no recognized roadmap for standardization in this area.

3.2.6 Distribution

An important industry trend is improved support for distribution (see Table 9). The goal is to make location transparent from applications. Stand alone distribution services are improving and becoming higher level. Where interprocess Remote Procedure Calls do not provide many guarantees (like "at most once" execution or security or synchronization), higher level distributed systems are providing more capability. Thus, OSF Distributed Computing Environment is a collection of distribution services that provides Kerberos-based authorization, a time service, directory services, and RPCs. OMG Object Request Broker provides an OO distribution framework for sending messages between distributed applications. DARPA ISIS, being developed at Cornell, provides several kinds of synchronization guarantees and replication in a distributed environment.

Table 9 Distribution

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
Object Management Group (OMG)	CORBA: Common Object Request Broker Architecture specification	Distribution	1993	Craig Thompson 214-995-0347 thompson@caci.com	Richard Soley (OMG) 508-820-4300 soley@omg.org	• 00 distribution framework • Multiple implementations from OMG member companies
Open Systems Foundation (OSF)	Distributed Computing Environment (DCE)	Collection of tools for distributed computing	1993	Drew Holland 214-575-2620	Bruce Huse bhuse@osf.org	• Kerberos security • Time service • Directory services • RPC • Distrib file mgr
DARPA/Cornell U.	ISIS Distributed Systems	Distribution, synchronization, replication	Now	Jose Blakeley 214-995-0362	Ken Birman (Cornell) 607-255-9199	• Distribution framework offering several kinds of synchronization (higher level guarantees than RPCs)

It is likely that the data base community will begin to build on higher level distribution services over the next several years instead of developing internal versions of these services. Experimental work in this area is in progress in the DARPA POB Open OODB project at Texas Instruments.

3.2.7 Generic Interchange Formats

Generically, "interchange formats" are domain-specific representations, or languages that are typically used to transport descriptions of design objects from one tool or environment to another tool or environment. They are usually used for bulk transfer of data. They may be in source (human-readable) or binary (more efficient) formats. Specific interchange formats exist for electronic design data (e.g., EDIF), documents (e.g., SGML and ODA), graphics (e.g., IGES), and CASE data (e.g., CDIF).

To date, there are hundreds, probably thousands, of interchange formats that are widely used. There are some standards in the area. ISO ASN.1 (Abstract Syntax Notation) is an international standard for a BNF-like generic interchange format construction capability (see Table 10).

Table 10 Generic Interchange Formats

COMPANY DOD PRODUCT STANDARDS ORG.	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY	TIME	TI POINT OF CONTACT	ORGANIZATION POINT OF CONTACT	COMMENTS
ISO ANS.1	Abstract Syntax Notation (ASN)	• RASSP interchange formats should be based on ASN.1	Now			• BNF like generic interchange format construction

Standards for many domain-specific interchange formats exist (e.g., for EDIF, IGES, SGML, ODA, etc.) and proprietary interchange formats abound (e.g., Maker Intermediate Format, or MIF, for Framemaker, etc.). The PDES/STEP community is using the object language Express as a specification language for interchange of mechanical CAD and more generally any product data. The CAD Framework Initiative (CFI) is developing a number of CFI "standard" representations for interchange.

Interchange formats differ from programs in that they do not necessarily contain control constructs though such "behavioral specifications" can be encoded, so there is a grey area. They differ from data bases in that data bases are under control of some DBMS system which permits selection operators on the data base to return subsets of the data encoded in the data base. They differ from class libraries in that they encode state information of instances representing designs, though they may also encode meta model data, for example, type data describing the structure of instances, so again this is a grey area. It is worth noting that there is a natural correspondence between information encoded in interchange formats and information encoded in class libraries and data bases, as shown in Figure 12.

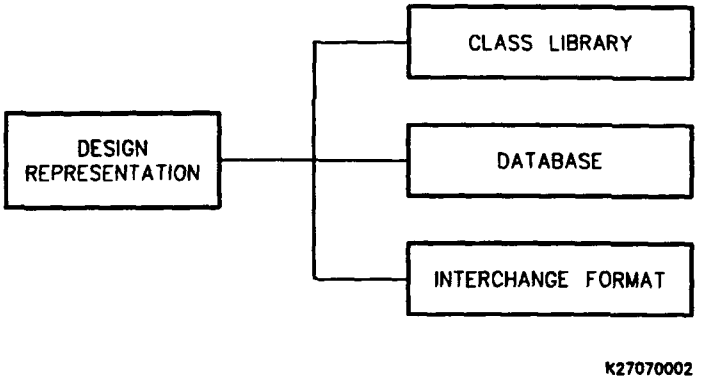


Figure 12
Automated Mappings Should Exist Between These Three Views of Design Representations

Today, in most cases, interchange format generators and parsers are manually developed, idiosyncratically for each interchange purpose. They are not generated from specifications. In addition, the mappings from interchange formats to data base objects and/or to class library objects are also manually developed and idiosyncratic.

The RASSP community will spend much of its time agreeing on common domain-specific representations that can be moved between tools and vendors. It would be a benefit if it could adopt one or a small number of common ways to encode and decode these domain-specific representations, since several will be developed. Since they can be encoded as interchange formats, it would be useful to develop or reuse a common "Interchange Format Generator Toolkit". The toolkit would contain parser-generator tools for automatically generating source and binary interchange formats from object representations. While the toolkit should be based on ASN.1, it should also remain open to allow non-compliant interchange formats to be registered with a common Externalization-Internalization Service. The Object Management Group (OMG) (see Section 3.2.3) has plans for such a service but no work is

planned until 1993-4. The RASSP community should work through the PDES/STEP and CAD Framework Initiative to put in place an Interchange Format Generator Toolkit, based on parser-generator and object technology. The DARPA Persistent Object Base (POB), DARPA Module Interconnect Formalism, DARPA DICE, and DARPA Knowledge Sharing communities should be involved in this effort.

3.3 Issues/Recommendations

This subsection highlights issues in the area of software infrastructure that the RASSP program will need to resolve. For each issue, we recommend an approach.

Issues 3.3.1 and 3.3.2 correspond to the issues shown in Table 11.
Issues 3.3.3 and 3.3.6 correspond to the issues shown in Table 12.
Issues 3.3.7 and 3.3.9 correspond to the issues shown in Table 13.

Table 11 Harmonization Needed - Object Models and Frameworks

ISSUE	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Babel of non-interoperable object models (e.g. C++ ≠ PDES Express ≠ SQL3 ≠ OMG IDL, etc)	Harmonization/convergence effort underway in ANSI X3H7	Highly desirable	More participation needed	ANSI X3H7 Object Model Interoperation	1992-1995	• Influence SQL3, PDES Express to participate in X3H7 • Fund R&D on object model interoperation
Competing/overlapping frameworks efforts - OMG, PCTE, CFI, PDES, EIS, ... Falcon, ... Need harmonization/convergence of object services and frameworks efforts	Harmonization/convergence effort just starting Standard APIs and interchange formats	Highly desirable	More participation needed	OMG, PCTE, CFI, PDES/STEP, EIS, Sematech,...	Earliest 1995	• Support Workshop on Application Integration Architectures (11/92) • Sponsor similar harmonization work
Roadblock to seamless integration: Commercial CAD frameworks (e.g. Mentor Falcon) are propretary/closed (may be technically open, not commercially); open systems frameworks (e.g. OMG) are immature	Industry effort to develop open frameworks underway. Use/influence emerging frameworks standards	Highly desirable	More participation needed	OMG, PCTE, CFI, PDES/STEP, EIS, Sematech, ...Mentor,...	Earliest 1995	• Influence CFI, CAx... to build on OMG • Influence Mentor Falcon, DEC Powerframe, ...to build on CFI • Develop standard API's and interchange formats • Insure CAD/frameworks are OODB independent (e.g. experiment--plug OODBs into Mentor Falcon, etc.)

3.3.1 Issue: Object Model Harmonization Needed

RASSP data representations will depend increasingly on object models. Currently, there are several competing object models that vary in semantics. During its life cycle, an object is designed using Object Analysis and Design tools, defined using an object model specification language like PDES/STEP Express, operated on in a programming language like C++, stored in a DBMS perhaps using SQL3's object model, transported across networks using OMG IDL/ORB object model, etc. At present, none of these object models is quite equivalent to the others. Programmer's must specify programmer-specific mappings across object model boundaries for each transition.

Table 12 Data Base Management

ISSUE	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Standard OODB API (interface) needed	Accelerate consensus leading to standards	Yes	Yes	ODMG, Open OODB, X3H2 SQL	1993-1995	• Encourage ODMG consortium • Fund precompetitive consortium for OODBs
Who will win---relational vs OODB?	Industry will develop hybrids; OODBs provide improved functionality and performance; queries still needed	Both	Yes	ODMG, Open OODB, X3 OODBTG, X3H2 SQL	1993-1995	• Encourage ODMG consortium • Fund precompetitive consortium for OODBs
Federated and decentralized RASSP repository/ DBMS needed	• RASSP data stored in files, RDBs, OODBs • Use DBMS standards • Use design representation standards • Open federated DBMS architecture • Open transaction protocol • Open query engine	Yes	More work needed	X3H2 | RDA, IBM DRDA, MMC Camot DARPA Persistent Object Base Program	1995	• Fund R&D on enterprise-wide info repositories, object file systems, distributed, WAN, and decentralized libraries, fine-grain change management, real-time • Identify and develop standards
Improvements to PDES/STEP Express	• Participation in PDES/STEP Express	Yes	More work needed	PDES/STEP Express, ANSI X3H2, ANSI X3H6, ANSI X3H7	1994	• Standard mapping to C++ • Queries for Express • Config. mgmt. for Express

Table 13 Interchange Formats and Libraries (Generic)

ISSUES	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Which interchange formats/domain representations to standardize; convergence of overlapping standards; inadequate standards	• Consensus: HIGH: PDES/STEP, VHDL... MEDIUM: rqmts spec. lang... LOW: ... • Focus on RASSP-specific interchange formats and representations	Yes	Yes—More participation needed	CAD will it fit? Framework Initiative, Atlas	1994-1995	• Accelerate harmonization efforts • Use RASSP as a pilot program to accelerate CAD representation standards
What to put in the RASSP library?	• Any reusable component	Yes	No formal approach	DARPA STARS Repository, MCC Electronic Databook	1993-1995	• Influence DSP vendors and CAD vendors to publish CAD libraries • Prototype RASSP reuse library in pilot programs
Bootstrap RASSP business	• RASSP libraries and services available to RASSP vendors • Amortize library reuse cost across organizations.	Yes	Yes—More participation needed	ANSI X3T3 Open Distributed Processing, DARPA MADE, MADE, MCC EINet, OMG	1994-1995	• Fund several RASSP efforts which must cooperate • Demo RASSP trader/broker based on OMG/ODP/EINet • Fund development of PDES/STEP standards for RASSP

Our recommendations:

- ANSI X3H7 is now in place to deal with object model harmonization and convergence issues. The RASSP working group should communicate requirements for interoperation of object model standards through communities like PDES/STEP and CAD Framework Initiative, which should in turn participate in X3H7.

- More research and development is needed in related areas: (1) mappings among object models; and (2) standard mappings for domain-specific interchange formats to "class libraries." This work should be assigned to or coordinated with the DARPA Persistent Object Base program.

3.3.2 Issue: Frameworks Harmonization Needed

There are many competing and overlapping frameworks. This is a roadblock to seamless integration. OMG and PCTE are generic frameworks. CFI and PDES/STEP are domain-specific frameworks. Mentor and Cadre are domain-specific and proprietary CAx frameworks. There is a need to harmonize and converge these efforts to preserve RASSP investments in tools and to provide portability. One puzzle is that proprietary frameworks are mature but commercially closed (though they may be technically open); open systems frameworks are less mature.

Our recommendations:

- Framework efforts (OMG, PCTE, CFI, PDES, Mentor, Cadre) should be influenced to be more coordinated. One way to do this is to participate in forums like the upcoming Workshop on Application Integration Architectures (planned for 1Q93), aimed at developing a roadmap for framework and standards convergence. The roadmap should identify how the outputs of one group can be the inputs to other groups, and should supply a schedule for when standards and products are going to be available. More efforts like this workshop are needed. For information on this workshop, contact Craig Thompson, Texas Instruments, one of the workshop organizers, or contact Gio Wiederhold, DARPA/SISTO.

- Specialized CAx frameworks (CFI, PDES/STEP, CDIF) should influence and build on generic frameworks like OMG and not re-invent whole CAx-specific frameworks. Proprietary frameworks (Mentor, Cadre) should be influenced to build on CAx-specific frameworks and generic frameworks. This is best accomplished by putting in place a RASSP technology transfer program that funds some cross-group liaison work and also by putting in place driving applications that require interoperation between these emerging standards.

  One form of communication between groups is negotiating on requirements and schedules. Another is to participate in the second group through change proposals.

- Standard APIs are needed for framework services (for example, change management). This requires participation (through CFI and PDES) in frameworks and services standards efforts.

- Frameworks should be OODB/DBMS independent for plug-and-play. This requires DBMS standard interfaces. See Issue 3.3.3.

3.3.3 Issue: The need for OODB Interface Standards

The OODB vendor community is addressing this problem, though with limited resources. As described in Section 3.2.5.2, Object Data Management Group (ODMG) is a consortium of OODB vendors that is working on OODB standards.

A DARPA-funded Open OODB Precompetitive Consortium would provide funds for ODMG and for OODB R&D. The DARPA Persistent Object Base program does the latter currently.

One project that would accelerate acceptance of an OODB standard would be a "call level interface" or similar product for the proposed ODMG interface. Each OODB company would supply mappings from the common interface to the functions supported by their products. Users would use the common interface and could then talk to any of the OODBs supported. This would help insulate applications from idiosyncratic changes to OODBs. A similar tool is being developed in the RDB community (see Section 3.2.5.4).

3.3.4 Issue: Who Will Win: OODBs or RDBs (Relational DBMSs)

This is really a non-issue. OODBs will have to add query capability to compete with RDBs; they are doing so. RDBs will have to add objects. This is the focus of ANSI X3H2's SQL3 standard, in development.

See Section 3.2.5.3 on hybrid RDB-OODB Data base Management Systems.

3.3.5 Issue: The Need for Federated, Decentralized, Scalable Intra- and Inter-enterprise-wide Repositories (Data Base Systems)

Research should be funded on open, extensible, scalable enterprise-wide information repositories and on federation, including work on object-file system hybrids, distributed data bases and WANs, decentralized libraries, fine-grained configuration management, and real-time data base systems. This research could be funded through the DARPA Persistent Object Base program.

See Section 3.2.5.4.

3.3.6 Issue: Improvements to PDES/Express are Needed

PDES/STEP is described in Section 3.2.4. The primary focus of the PDES community should be in standardizing cross-industry domain-specific representations (entities in important domains of discourse).

The PDES community has developed the Express object model "specification language". One standard mapping from Express to C++ (that covers all of Express) is needed. This effort is ongoing but should be accelerated. This community should participate in ANSI X3H7, which is working on object model harmonization.

Also, a Query capability for Express is needed. The Express query language should be based on SQL but use the Express object model.

3.3.7 Issue: Which Interchange Formats and Representations to Standardize?

Which interchange formats and representations to standardize? How to avoid overlapping and inadequate standards? Where to avoid standards?

A RASSP Working Group needs to identify key potential standards and rate them for importance and potential of consensus (high, medium, and low). The group should work through larger efforts like CFI and PDES/STEP to develop draft standards and improve existing standards where needed. In this way, RASSP can act as a pilot program to accelerate CAx representation standards.

3.3.8 Issue: What to put in the RASSP Library?

The simple answer is, anything that is potentially reusable. It is likely that experimentation will be needed to determine the answer.

Experiments are needed to determine whether only components engineered for reuse are in the repository or also whole past designs.

A Module Description Language may be required (possibly based on DARPA MIF) to generically describe library elements. DARPA STARS Repository or MCC Electronic Databook projects may also provide similar library descriptor technologies. TI Prism library technology, like the proposed RASSP approach, is based on a library approach and is another source.

Access to the repository needs to be content-addressable as well as navigation and query based.

The RASSP program needs to address incentives for multiple vendors to put designs in a common (logically common, possibly physically distributed) repository. One way to do this is to require RASSP vendors to work together on a common family of RASSP designs using and developing a common RASSP design library.

3.3.9 Issue: How to Bootstrap the RASSP Business?

A decentralized repository/library that crosses organization boundaries provides the best chance of amortizing cost of developing DSP libraries. This implies development of a services and components trading function to develop world-wide markets via "virtual companies" using ideas like those in the "21st Century Manufacturing Enterprise Strategy" report and facilities like that of MCC EINet or ANSI X3T3 Open Distributed Processing. See 3.2.1 Enterprise Integration.

3.4 Summary of Key Recommendations

The RASSP community should adopt and develop RASSP specific tools, representations, and standards; but it also need to influence development of tools, representations, and standards that it needs but that are being developed for use by broader software infrastructure communities.

Some of the recommendations made in this section require RASSP-specific funding; some require strong liaisons and cooperative demonstrations between the RASSP program and some of the software infrastructure communities or some other DARPA programs.

- The RASSP program should be structured to have a major technology transfer component overseen by a RASSP Working Group (steering committee or consortium). The RASSP Working Group should establish an Infrastructure Team whose mission is to plan out how to leverage other industry groups to accelerate consensus that leads to

standards in areas critical to RASSP. This will involve (1) development of a focused and common RASSP vision, (2) identification of a rank ordered list of goals and milestones, targeting key standards to be developed or refined that RASSP will depend most heavily on, (3) building strong liaisons with key standards groups, industry consortia, and key individuals critical in developing the software infrastructure that RASSP will need, (4) identifying key other DARPA communities (DSSA, DICE, MADE, POB), and (5) developing a technology transfer and demonstration plan for putting RASSP capability in place.

- RASSP should plan to identify key frameworks (we recommend OMG, CFI, and PDES) and focus on working through these groups to insure its more generic framework needs will be met. In turn, RASSP should lobby these groups to resolve issues like selection of a common object model (through participation in ANSI X3H7 Object Models) and harmonization of object service framework architectures (probably through participation in ANSI X3H4.1 and X3H4.2 Repository Architectures and Services). This may involve working through MCC's CAx Consortium and may also involve working with other groups. A RASSP advantage in working through CAD Framework Initiative and PDES/STEP is that it can afford to narrow its focus to exactly signal processors and thus focus some CFI effort on accelerating what is needed to make this area successful.

- DARPA/ESTO should build a liaison with DARPA Persistent Object Base program and lobby it to fund an Open OODB Precompetitive Consortium or similar program, whose members are the DARPA POB projects, the OODBMS companies and perhaps some SQL3 vendors. The purpose would be to collectively develop OODB application program interface (API) standards and also to develop, or fund development of, common modules (e.g., storage engines, query engines, distribution engines, etc) that will progress the capabilities of the entire OODB community. Another purpose would be to encourage OODB and RDB interface standards convergence, perhaps by building an ODBC-like call level interface for the ODMG specification. By architecting a RASSP system to be specific OODB independent and by adopting emerging OODB standards, RASSP will be able to remain specific OODB independent to a large extent.

- Through a liaison with DARPA/SISTO, DARPA/ESTO can also support projects aimed at developing scaled up, open data base system \architectures, especially on federated data base systems and object-file system hybrids. This could be accomplished through cooperation with the DARPA Persistent Object Base community.

- The RASSP community should develop or identify a standard and generic Interchange Format Generator Toolkit to automatically generate parser-generators for new domains from specifications. Work on this should be in conjunction with the CAD Framework Initiative and PDES/STEP communities, possibly by jointly funding such work through the DARPA Persistent Object Base or Module Interconnect Formalism communities.

- DARPA should encourage development of a common RASSP reuse library by require RASSP vendors to work together on a common family of RASSP designs using and developing a common RASSP design library. Library elements and common services should be available at remote sites.

- RASSP should avoid funding significant work on proprietary infrastructure technologies.

4. APPLICATION DESIGN SYSTEM

By System Design, we mean the integrated design and analysis of both the hardware and software components of a complex military system, and by system, we are here restricting our attention to computer-based signal processing systems.

Figure 13 illustrates a typical breakdown of the major activity stages in a System Design process as it is generally practiced today. Above each stage name are shown some of the major data bases or data elements that are generally employed in that stage. Below each stage name are shown some of the major design activities and design tools generally employed in that stage. These lists are not purported to be complete in any sense, but they do show some of the typical data bases required and design activities and tools required for each of the stages in the System Design process, and together they serve to illustrate a common view of the System Design process. Unfortunately, the System Design process is at present a completely informal process, and even when it is consciously followed, it is usually followed in a thoroughly ad hoc manner.

Fortunately, many of these tasks are quite difficult, and many of the required data elements are quite complex. Powerful tools and technologies have implemented on the system design process. But unfortunately, these tools and technologies have not yet been brought to any sort of formal integration that allow a systems design engineer to explore multiple facets of a complex system design from a variety of simulation and modeling approaches, and to be able to transfer design decisions and insights from one design activity to another. Further,

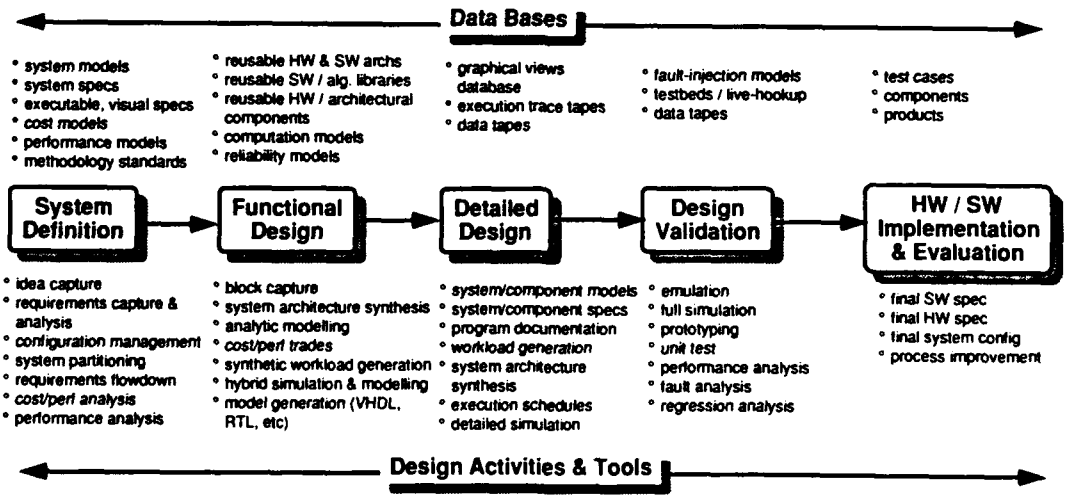


Figure 13
Typical Breakdown of Major Activity Stages

little capability exists for taking a system design process from the point of formal specifications and system requirements through a detailed analysis, modeling, simulation, and design activity and then to a formal set of design specifications for software and hardware implementation.

For RASSP to succeed, such integration is both necessary and desirable. Our approach to this high level of integration is described in the following sections. The main theme underlying our approach is the reuse of both formal software reference architectures and hardware reference architectures through Domain-Specific Software Architectures (DSSAs) and Domain-Specific Hardware Architectures (DSHAs). These architectures are integrated through a formal System Design Environment coupled to knowledge-based design advisors and shared Information Object Modelers (IOMs).

Additionally, we believe the System Design process must be formally defined and modelled if the goals and needs of the RASSP methodology are to be met in the system design arena. This formal process must support both in-cycle and out-of-cycle system design activities. By "in cycle" design activities we mean those design activities initiated as a direct result of a new product or model-year upgrade procurement process. By "out of cycle" design activities we mean those design activities that lead to the reuse of, modification of, and extensions of both the software and hardware reference architectures and the System Design Environment in preparation for use in "in cycle" design activities. We further believe that our approach will and must support multiple modes of System Design activities, as we recognize that not all design activities of value to RASSP will be carried out as a result of formal product or model-year upgrade procurements. Other modes of system design to be supported include managed experimentation, on-going analysis, design upgrade, and continual "what if" explorations.

- In-Cycle Support:
  - Assumes reuse of HW/SW architectures (extended and integrated DSSAs and DSHAs)
- Out-of-Cycle Support:
  - Assumes reuse of, modifications of, and extensions of HW/SW architecture, as well as introduction of new HW/SW architectures
  - Allows "what if" analysis based on other-than-production goals
- Modes:
  - Production Design, Model-Year upgrade, experimentation, on-going analysis and design upgrade.

Finally, we believe the System Design process must be formally modelled to the extent of supporting multiple, well-defined measurement points and metrics for use in on-going benchmarking activities to support the RASSP methodology.

4.1 DSSA Software/DSHA Hardware Development

Figure 14 presents a possible top-level view of a formal System Design process capable of supporting both in-cycle and out-of-cycle system design activities, as well as supporting multiple modes of system design activities. The process model integrates both a DSSA-based software and a DSHA-based hardware design process. The process can begin from multiple points, depending on the goals of the person engaged in a particular system design effort. Each goal prescribes what in the previous section was called a "mode". Here, the modes shown include Managed Experiments, formal requirements (for both new product and model-year upgrade design), and "what if" analysis. As the process proceeds, both application analysis and domain analysis are required and

supported, as we realize the possibility of either software or hardware designs that truly cannot be effectively generalized into a particular domain. Our belief is that the greatest advances in cycle-time reduction will result from reuse of domain-specific designs and knowledge, but even when such domain-specific data is absent,

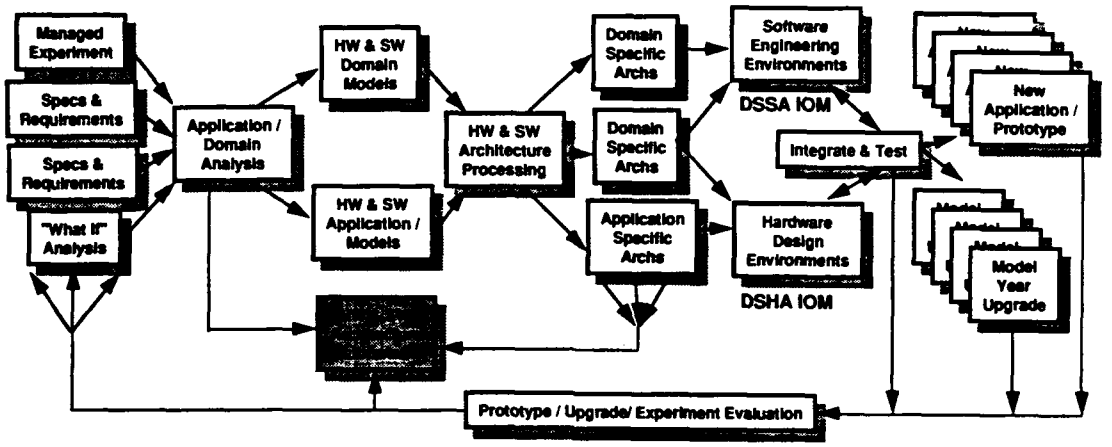


Figure 14
Possible Top-level View of Formal System Design Process

application-specific designs and knowledge can and should be directly supported by the DSSA and DSHA Development Environments. To simplify the remainder of this part of our presentation, we restrict our discussion to domain-specific design issues.

The description of the activities engaged in during each step of the process depends on whether we consider the design and development of a new system or whether we consider the redesign, modification, or extension of an existing system that has already been through the process, as well as whether we consider the design activity to be "in-cycle" or "out-of-cycle" activity. In any event, depending on the goals and mode of the design activity, the system requirements are analyzed from a domain or application viewpoint, the hardware and software architectures are formalized and made reusable and extensible, and domain knowledge and design decisions are recorded for future and on-going design activities. These architectures, together with the domain and application related knowledge, data, models, and decisions are then made available through an integrated Information Object Modeler for access by both the software and hardware design environments.

Hardware and software designs and/or implementations can then be iteratively brought together and integrated for formal product design or model year upgrade, including final integration and testing. Additionally, as can be seen in the process flow, if the activation mode were one of managed experiment, prototype analysis, or "what if"

analysis, reflections on the results of the experiment or design change can simply be fed back to the libraries, back to the specifications, or back to the experimental environment.

The process model depends on and benefits from domain-specificity in the system design process in two ways. First, by narrowing our attention to specific domains in which both significant reuse of both a software reference architecture and a hardware reference architecture is exploited, many seemingly unbounded problems become bounded and thus manageable and possibly automatable. The restriction to certain, specific domains also benefits the overall system design process, process model, and RASSP goals by making possible the development of intelligent or knowledge-based force-multiplication technology and approaches in the supporting system design tools and system design environment. It is perhaps these domain-specific force multipliers that will most support the short design-time goals of the RASSP program. Thus in our view, the term "domain specific" refers less to a diminution of capabilities than to a significant increase in capabilities that would not otherwise be achievable in a general-purpose approach.

There is currently a DARPA-supported DSSA technology development effort underway. The DSSA program is headed by Lt. Erik Metalla of DARPA SSTO. The current level of funding is approximately $25M over four years (1991-1994). Six teams are currently involved in the DSSA effort: (1) IBM, (2) GTE/Comtel, (3) Teknowledge, (4) ORA, (5) Honeywell, and (6) TRW. In addition to these teams, a number of other companies, universities, and research centers actively track the DSSA program and attend the progress meetings. While there is no equivalent DSHA program underway, we believe many of the technologies being developed by the DARPA DSSA effort will prove immediately applicable to supporting our vision of a DSHA. Further, many ongoing research and development activities in both industry and academia will admirably fit into and support the goals of a DSHA. The development of a DSHA and its supporting Information Object Modeler should become a formal part of any RASSP effort.

In addition to the DSSA and DSHA development activities, a number of other technologies and programs that will prove beneficial to other integrated DSSA/DSHA System Design visions are being pursued; some of these are listed in Table 14. Many of these will play a vital role in achieving the TI RASSP vision.

Table 14. Other Technologies Relevant to DSSAs/DSHAs

Relevant Technologies/Programs
STARS, Arcadia, Prototech, MARVEL, etc.
OODB (DARPA Open OODB, Falcon)
Knowledge DBs (Stanford, ISI, CMU)
Visual Design Envs. (ONR, FORGE,PieScope)
Architectural Synthesis (Micon, ViParSim, Ptolemy)
Massively Parallel Sim. (HPCC, PADS, VHDL)
Hierarchical Design Advisors (MCC, CMU, SES)
System-level Simulation (Ptolemy, ADAS, SES)

4.2 ASSP Application Design System

Figure 15 shows a top-level view of our proposed RASSP Application Design System. The Design System relies upon and exploits the reusable software reference architectures and reusable hardware reference architectures provided through the DSSA and the DSHA Information Object Modelers. The "reference architectures" are those software and hardware architectures that represent the product or system designed during out-of-cycle design activities and are being upgraded, analyzed, or experimented with, either in a procurement-driven in-cycle design activity. For an out-of-cycle design activity; for ease of discussion, the reference architectures are also referred to as the DSSAs or the DSHAs themselves, although such usage of these terms is technically ambiguous.

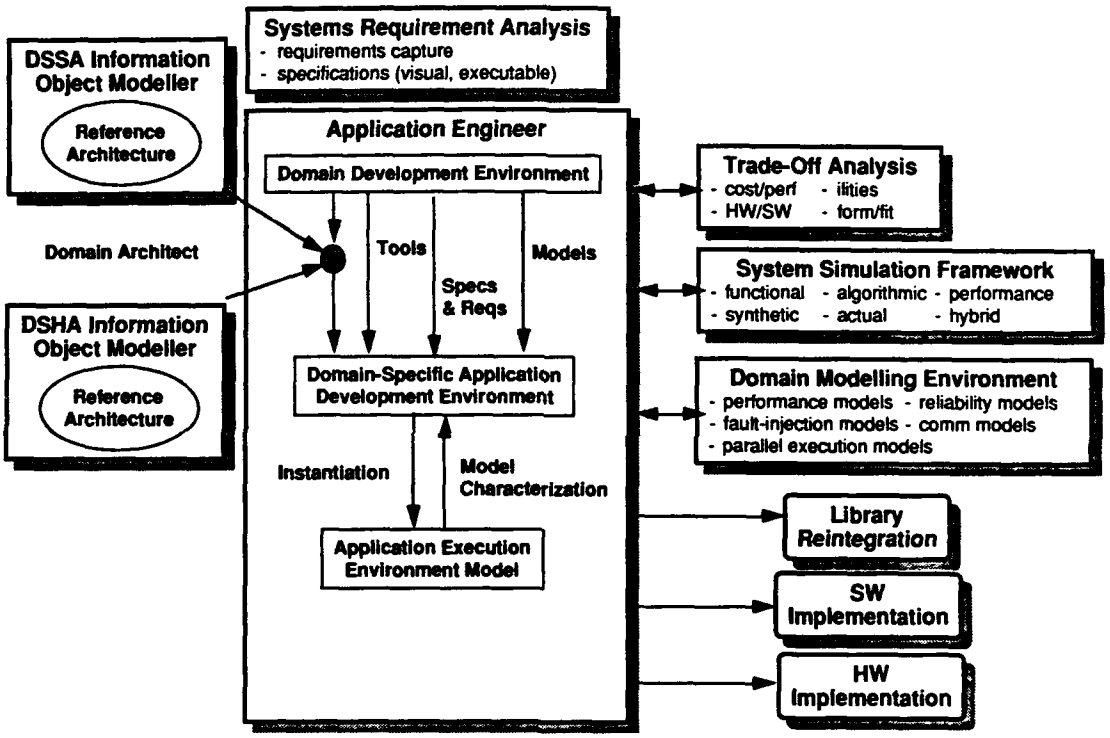


Figure 15
Top-level View of Proposed RASSP Design System

The responsibility of defining, maintaining, and extending the DSSAs and the DSHAs, along with their supporting Information Object Modelers, falls to the persons called the Domain Architects. The Domain Architects, mostly through out-of-cycle design activities, support the usage of the DSSAs and the DSHAs by the Application System Engineer. It is the Application Engineer who, during an in-cycle design activity, is responsible for taking a set of formal system specifications and requirements and delivering a formal, integrated system design that meets those specifications and requirements. Within the RASSP Methodology, this is done through the modification and extension of reusable hardware and software domain-specific, reference architectures. Although the DSSAs and the DSHAs are maintained and supported separately, they are integrated during application engineering through

software-to-hardware mappers and machine organization synthesis. The Domain Development Environment consists of a suite of integrated modeling, simulation, testing, analysis, and synthesis tools that are used both to integrate the DSSAs and the DSHAs and to perform a variety of managed experiments and benchmarking trials. These exploit multiple formal system domain models such as performance models, cost models, and reliability models.

As the domain-specific application is developed, the Application System Engineer also relies upon a model of the Application Execution Environment. Instantiated models of the system design are fed to the application execution environment for validation, testing, and analysis, with formal characterizations of satisfiable application execution environments feeding back to the Application Development Environment. When the Application System Engineer is satisfied with the design, tests, and analyses, the completed design can lead to either implementation followed by test and integration, or simply back to DSSA, DSHA, and library re-integration, depending on the mode of the design activities being pursued.

Note again that the view presented here supports both in-cycle and out-of-cycle system design activities, as well as the multiple usage modes.

4.3 Design Integration and Validation

Given an existing DSSA and DSHA Information Object Modeler, the Application Engineer is responsible for integrating the software architecture and the hardware architecture and validating the resulting integration against the system specifications and requirements. To perform this integration and validation, the Application Engineer must instantiate both the software architecture and the hardware architecture. *By suitably parameterizing* these two architectures according to the required performance, cost, size, power, and reliability attributes, the Application Engineer initiates a set of bindings of code and hardware modules to the architectural components of the DSSA and the DSHA, respectively. A software model configurer and a hardware model configurer are controlled and guided by the Application Engineer to accomplish a set of bindings that satisfy a set of design constraints and criteria. The constraints and criteria define the set of acceptable software-to-hardware mappings, execution timing schedules, and data access behaviors for the intended system.

Given the two instantiated configurations, the Application Engineer can then simulate and model various aspects of the integrated system design. Through such simulation and modeling, the Application Engineer can perform a variety of hardware/software tradeoffs; analyze multiple real-time schedules for robustness; assess the suitability of alternative hardware modules, memory configurations, and bus interconnections; explore alternative hardware configurations; and determine the optimal mappings of the instantiated software system to the instantiated hardware system.

Today much of this sort of analysis and exploration is done by hand, with very generic tools that know little or nothing about the functional responsibilities of the system being designed/examined and little or nothing about the environment within which the system will be fielded. Consequently, while the Application System Engineer may receive productivity support through the use of these tools, little cycle-time reduction results since the tools can offer little, if any, analysis of the results of a trade-off study or a particular software-to-hardware mapping approach, for example. Consequently, when humans are heavily involved, rapid design frequently does not imply optimal design.

By focusing on specific application domains and relying on the significant reuse of both hardware and software reference architectures, we believe many design support tools can be extended into the domain-analysis arena and provide knowledge-based, application-specific design advise and analysis. Eventually, this extension of design tools into the intelligent, domain-oriented arena will allow more and more of the System Design process to become automated. Eventually, an automated design environment will be developed in which the Application System Engineer provides minimal correction and advice to an automated design system, rather than as today, where a set of semi-intelligent design advisory tools give minimal advice and correction to the human system designer. We want to reverse the roles!

To assist in the automated design process, the Application Engineer must be presented with a variety of domain-independent and domain-specific animated, graphical presentations of the various aspects of the system's design, construction, and behavior. These multiple, animated, graphical views are actually part of the Domain-Specific Application Development Environment's set of predefined graphical support system. The views provide both quantitative and qualitative animated views, with full support for virtual or synthetic reality explorations of complex, time-dependent behaviors and interactions. Because much of the design process will have been abstracted, encapsulated into domain-specific design advisors, and automated, a single Application Engineer can assume responsibility for multiple areas of a system design that are today addressed by multiple design groups, each consisting of multiple people, including hardware fabrication, cost estimation, cost and performance tradeoffs, and production scheduling. This will be made possible through an integrated, design-by-visualization application design environment.

In the process illustrated in Figure 16, following completion of the *simulation, modeling, and testing* design activities, the resulting formal designs, specifications, and requirements are used to implement the software and hardware components, or again, simply to feed back into the Libraries or DSSA/DSHA Information Object Modelers.

4.4 Extended DSSA Support Environment

Figure 17 illustrates some of the fundamental components of, and access methods defined for, an Extended DSSA Support Environment. The majority of the domain-specific models, knowledge bases, data repositories, design advisors, and architecture components are integrated through the DSSA Information Object Modeler. As described, the DSSA approach to reusable software benefits from domain specificity in two ways. First, by being able to restrict its attention to specific domains, the problem space becomes manageably small. Problems that first appear beyond our present capabilities suddenly become approachable, thus leading to early application of leading-edge technologies, albeit in limited (domain-specific) arenas. Second, by allowing currently tractable technologies to be extended in unique and force-multiplying manners significant capabilities for automated design that would be impossible in the domain-independent arena can be achieved. Such domain-specific extensions and multipliers are encapsulated within the tool set and supporting data domains for each particular DSSA. These include simulation models for the multiple functions of the DSSA, domain-specific reliability, performance, and execution models, among others, libraries of test data, workload suites, execution traces (both synthetic and real), software synthesis models, and the core of the reusable software libraries themselves.

Associated with the DSSA Information Object Modeler are a number of support tools, including the simulation systems themselves (not the

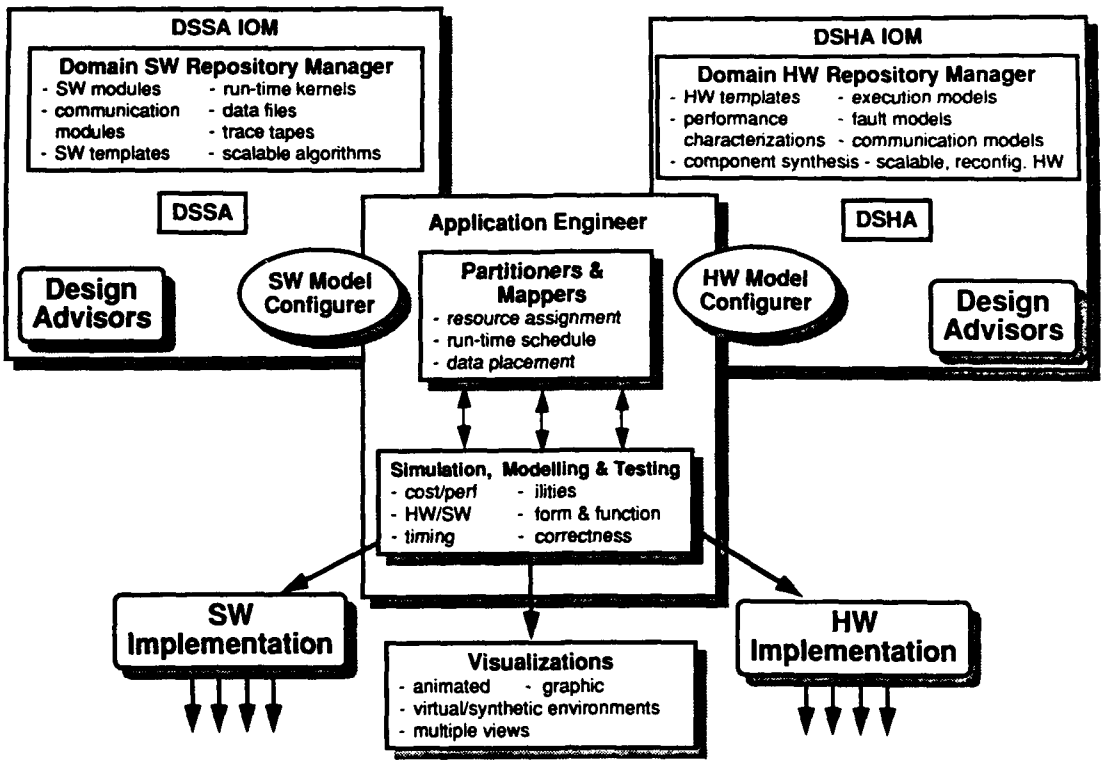


Figure 16
Design Integration and Validation

simulation models), parameterizable software partitioning and mapping tools, domain-knowledgeable, off-line, run-time schedule generators, and complete software code generation and synthesis tools capable of extending scalable, parallel algorithms to fit application-specific configurations and performance requirements. We view these tools as being domain-independent (even general purpose), but capable of supporting domain-specific applications and knowledge through parameterized executions, where the parameters are domain-specific and are provided by the DSSA IOM. An analogy can be seen with knowledge-based expert systems; these consist of a domain-independent inferencing mechanism and a set of domain-specific, knowledg encapsulating rules and theories. While the inferencing mec nism can support a large number of rule bases, it is the rule bases themselves that incorporate the domain-specific knowledge and make possible the design, synthesis, and analysis of complex system application. Accordingly, the DSSA IOM can benefit from broad-based, domain-independent technology advances in the general sciences of simulation, modeling, partitioning and mapping, and performance prediction.

While the view of an entire extended DSSA Support Environment and DSSA Information Object Modeler presented here may be beyond the scale of achievability within the 4-year RASSP Implementation program, it is important to realize that even without this view, some overall organizational philosophy of software system architecture and software reuse is required to make even the first level of the RASSP methodology realizable. To us, this first level of organization is respectably supplied by a minimal DSSA technology. The good news is that the DSSA technology is currently being developed under DARPA guidance, and it is

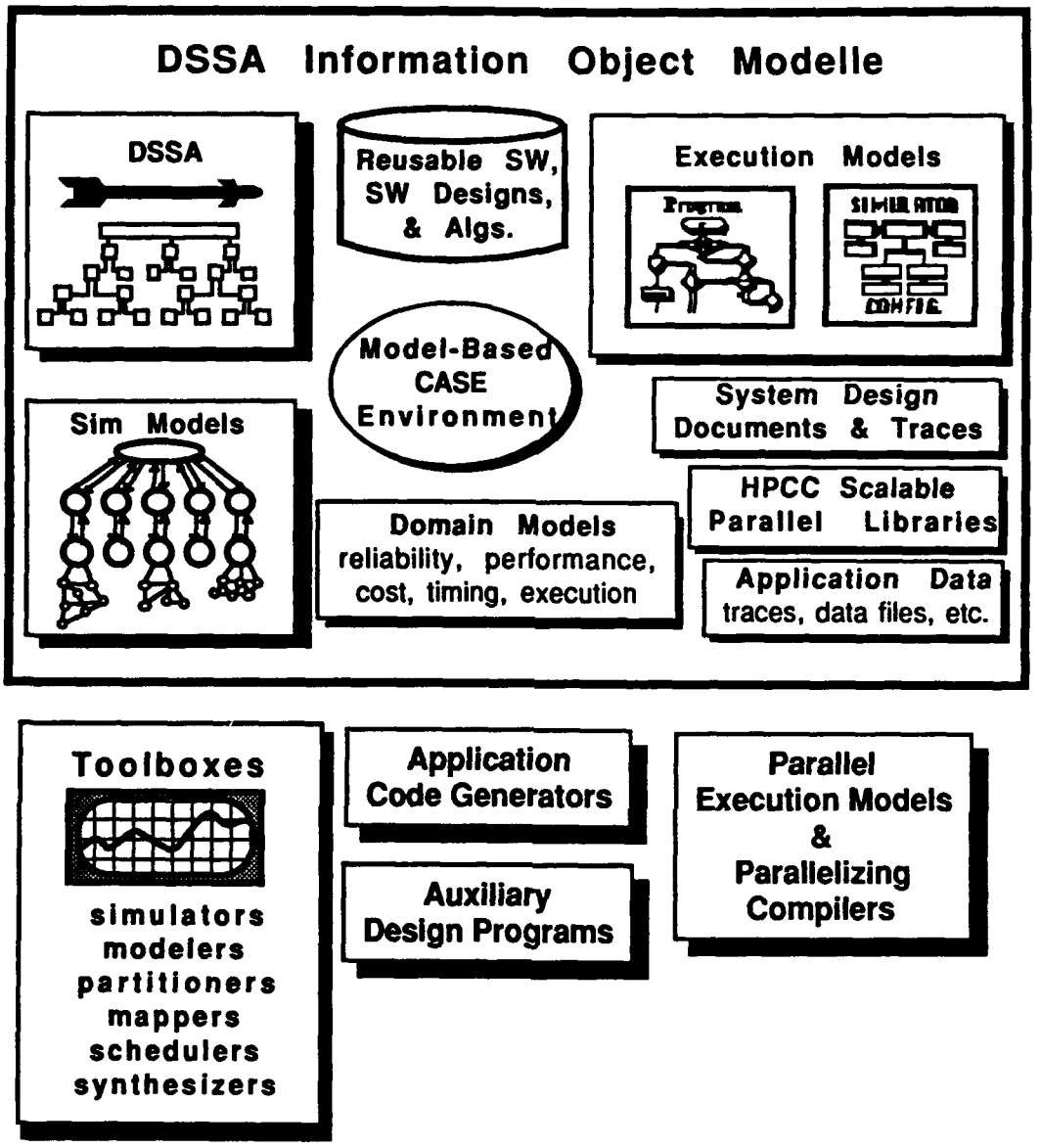


Figure 17
Fundamental Components and Access Methods

being developed within the right time-frame. The DSSA technology being developed can easily be extended to support the rapid prototyping and design requirements of RASSP and to support a formal model year upgrade process and process model.

Because the DSSA technology will evolve over time, we have presented our view of where this evolution will lead. We also believe the DSSA approach should be extended to contemplate an equivalent DSHA technology, and particularly one in which multiple, parallel processors play a leading role in future military system designs. The DSSA and the accompanying DSSA IOM must be integrated into the formal System Design

process and process model. Finally, the following lists includes some of the most important needs for an effective realization of the RASSP Methodology based on the DSSA approach.

- DARPA DSSA methodology
  - Extend to accommodate DSHA
  - Extend to accommodate parallel processing
- Domain-Specific Design Advisors
  - System: software partitioning and mapping, performance, real-time schedule, data placement, etc.
  - Application: guidance and control, parallel image processing subsystems, etc.
- DARPA open OODB technology
  - Extend to high-performance Knowledge DB
  - Extend to Active DB with "design interest" activations and process enactions
- Common SW system simulation model
  - Timing, function, performance, reliability, etc.
  - Analytical modelling, discrete-event, Petri-net, etc.
  - Common interfaces / data exchange
  - Hybrid; hierarchical; animated simulation views
- Formal system-level specification languages
  - SW architecture description languages
  - Performance specification languages
  - Temporal behavior specification languages
  - Fault-behavior specification languages
- Parallel execution traces & data models
  - Methodologies for capturing parallel executions
  - Synthetic workload generators / trace tapes
- Application and process benchmarks
  - DARPA Image Understanding Benchmark
  - ISPW-6 SW process model; common sensor data files

4.5 Extended DSHA Support Environment

The DSSA technology currently being developed to support reuse of software reference architectures can be easily, directly, and naturally extended to support the reuse of hardware reference architectures, and within the RASSP methodology. Reuse of hardware reference architectures is fundamental to the goal of rapid, model year upgrades for complex military systems. This ability would be provided through an integrated Domain-Specific Hardware Architecture (DSHA) System Design Environment. Figure 18 illustrates some of the fundamental components of and access methods defined for an Extended DSHA Support Environment. The majority of the domain-specific models, knowledge bases, and architecture components are integrated through the DSHA Information Object Modeler. Just as software architectures can be elevated from the application-specific level to the domain-specific level, we believe hardware architectures can be extracted from existing, fielded designs, that they can also be elevated to the domain level. Further, we believe that hardware architectures can be reused for new system designs as well as for model year upgrade. Texas Instruments, for example, has a well-defined, formal architecture for guidance and control sections of missiles that has seen significant reuse. Further, our own internal research with the DSHA approach has led to significant insights and substantiations of these beliefs.

As with the DSSA Information Object Modeler, the DSHA Information Object Modeler contains domain-specific extensions and multipliers encapsulated within the tool set and supporting data domains for each particular DSHA including, for example, detailed, hierarchical and hybrid simulations for hardware modules at the system, chassis, board, multichip module, and component levels; parallel architectural models represented in

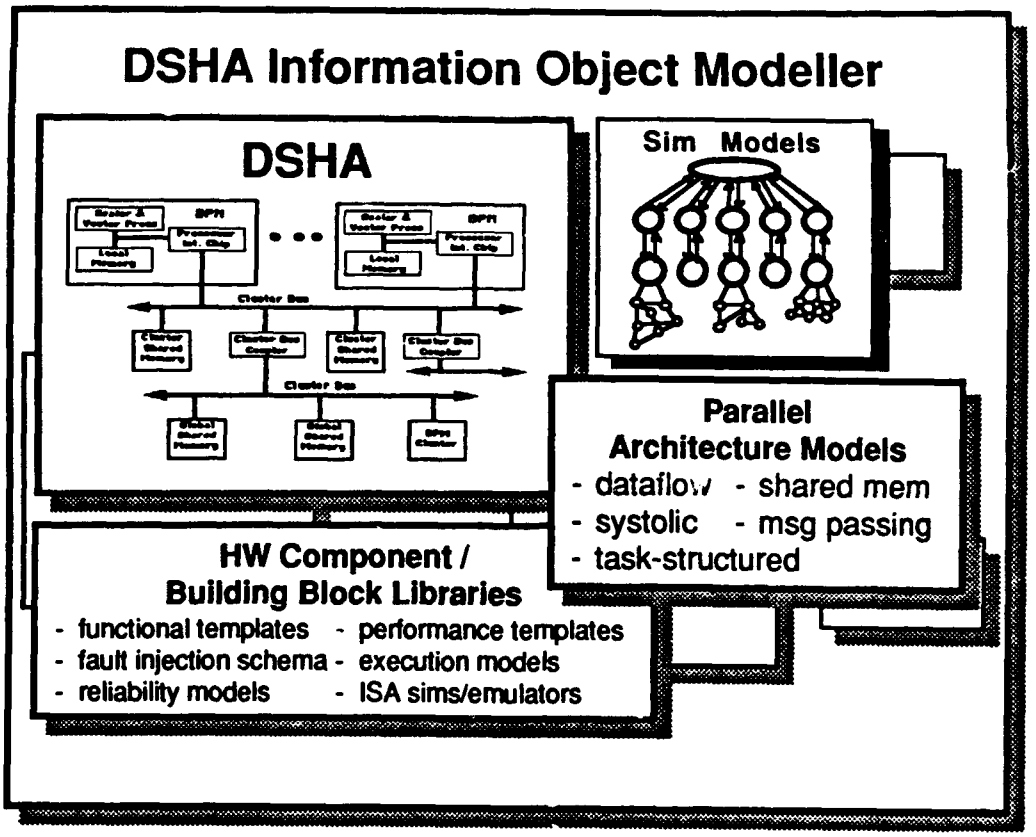


Figure 18
Fundamental Components of Access Methods for DSHA Support

multiple performance, behavioral, and costs aspects, and in multiple simulation languages and abstractions; and hardware building block and component libraries, with hardware templates that allow extraction-by-need of performance, execution, or reliability data, models, and/or specifications.

As with the Extended DSSA Support Environment, the Extended DSHA Support Environment is associated with a number of support tools, including simulation systems, hardware partitioning and synthesis tools, synthetic workload generators, and fault exercisers.

The following list shows some of the most important needs for an effective realization of the RASSP Methodology based on the DSHA approach; this list certainly does not claim to be exhaustive. While a DSHA will differ significantly from a DSSA, we believe much of the underlying technology development can be shared and mutually exploited by both a DSSA and a DSHA development program. As with the DSSA, the DSHA and the accompanying DSHA IOM must be integrated into the formal System Design process and process model.

- Adapt DARPA DSSA methodology to DSHA
- Domain-Specific Design Advisors
  - Packaging, system synthesis, component synthesis, HPCC architectures, reliability
- DARPA OODB technology
  - Extend to Knowledge DB

- Extend with Active DB with "design interest" activations
- Architectural description language(s)
  - Domain-specific module interconnect language
  - HW module/component/system arch descriptions
  - Domain-specific parameters/generators
  - Performance reliability characterization language
- Simulation models
  - Model-based analytic models
  - Parallel discrete-event simulation systems
  - Plug-compatible interfaces for hybrid simulation
- Architectural synthesis
  - Library of scalable, reconfigurable building blocks
  - Algorithm-to-silicon compilation
  - Application-specific adaptation of reusable HW arch
  - Scalable reliability
- Flexible memory architecture
  - shared memory, cache-coherent, physically distributed; real-time compatible
- Experiment management system
  - Stimulus generation, monitoring, data collection and analysis
- Software to hardware mapping
  - Simulated annealing / genetic algorithms, dynamic load balancers, data mappers, real-time schedulers (off-line & dynamic)
- Visual modelling & simulation tools
  - Qualitative as well as quantitative graphical views
  - Animated virtual world presentations of complex cost/perf and execution data models

4.6 DSSAs/DSHAs

Figure 19 illustrates three example DSSA reference architectures down the left column and three example DSHA reference architectures down the right. The top, left example is from the domain of "ground vehicles". This example was used in a recent DARPA-sponsored workshop on DSSAs. The second DSSA example is of a missile seeker. It too was presented at the DARPA DSSA workshop. Note that it is written in the Object-Connection-Update formalism. The bottom DSSA example is of a missile guidance system developed at TI as part of the RASSP study phase effort, but used in several DoD projects throughout the past several years.

The three DSHA examples on the right illustrate, respectively, a high-level breakdown of the hardware architecture of a signal processing system, a missile signal processing system, and a reconfigurable, scalable hardware architecture capable of supporting extremely high-performance, missile-based, signal processing systems.

In the middle of this chart are two illustrations that demonstrate the integration of both a DSSA and DSHA Information Object Modeler through an Application Development Environment. These illustrations also show the ability of the ICMs and the Application Development Environment to support multiple graphical views, each based on the differing information and modeling needs and expertises of the multiple users of the architectures. What is important to us is to provide both an integrated set of views that can be used by the different people involved in a RASSP effort, and to present the different aspects of a design and implementation effort to a single person, letting that person explore the ramifications of his/her design decisions on other parts of the product effort in ways that are not currently possible.

Because the types of views needed to present different information flows, design parameterizations and instantiations, and cost/performance aspects, for example, all require differing models of visualization, it

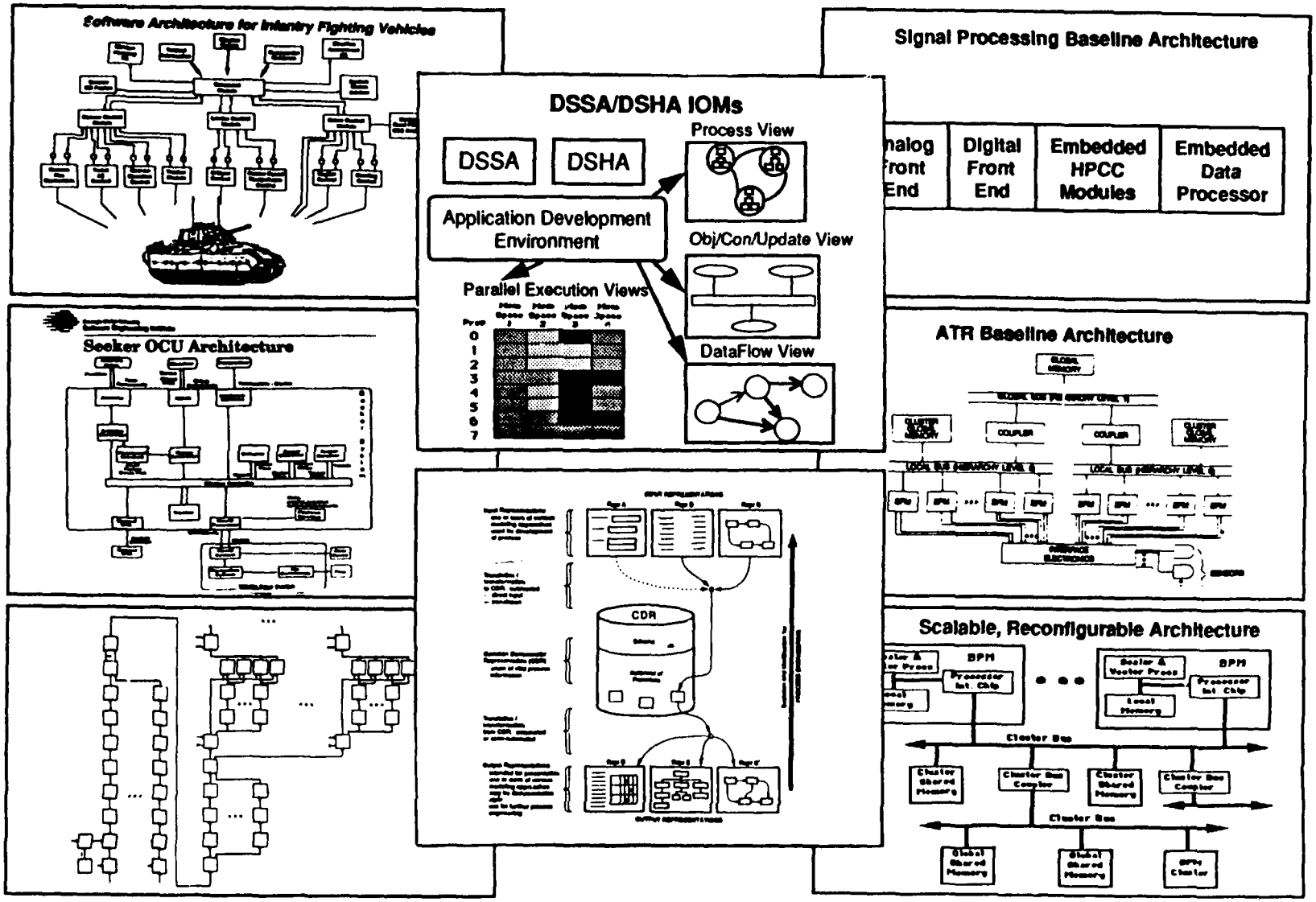


Figure 19
Three examples DSSA Architectures

is important to be able to translate information stored in one model into information for another model. We believe it is both necessary and important to develop a formal Application Design System that utilizes the technology of Common Denominator Representation concepts to support multiple, wide-ranging modeling objectives by the multiple users over a variety of usage modes, from model-year upgrade, to initial system-design, to managed experimentation. These representations must be developed as part of the overall RASSP program effort.

5. SOFTWARE

5.1 Overview

Within a system development, software has become a driving factor. Compared with systems of the past, current and future systems rely on a greater percentage of the system being programmable (more software to develop) and increased complexity of the functionality being programmed (more complicated applications, more complicated communications in multi-processor systems, security requirements, fault tolerance requirements, etc.). The DoD is well aware of "the Software Problem" and is taking it seriously. The draft DoD Software Technology Strategy (SWTS) addresses the many aspects of the problem, identifies work underway in various areas, and outlines plans for software technology investment to achieve its year-2000 objectives. These objectives are in line with the requirements for software development in the RASSP vision.

To support software development as defined in the RASSP vision, a combination of "work avoidance" (reuse), "working smarter" (process), and "working faster" (tools) will be required for both the in-cycle rapid-prototyping development and the out-of-cycle software library development and maintenance. Figure 20 shows the interfaces between the system design activity, the software library and the software engineering environment. The in-cycle software design activities are supported primarily by retrieval of existing components from the software library. The software engineering environment supports the out-of-cycle software development and those in-cycle developments required when a needed component is not contained in the library.

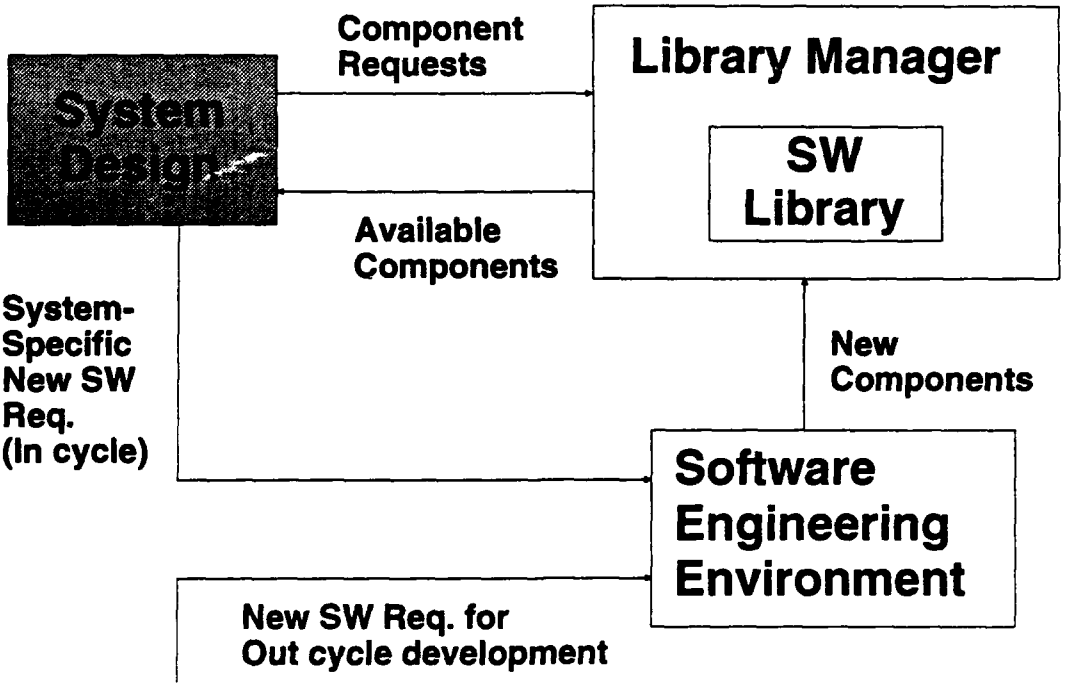
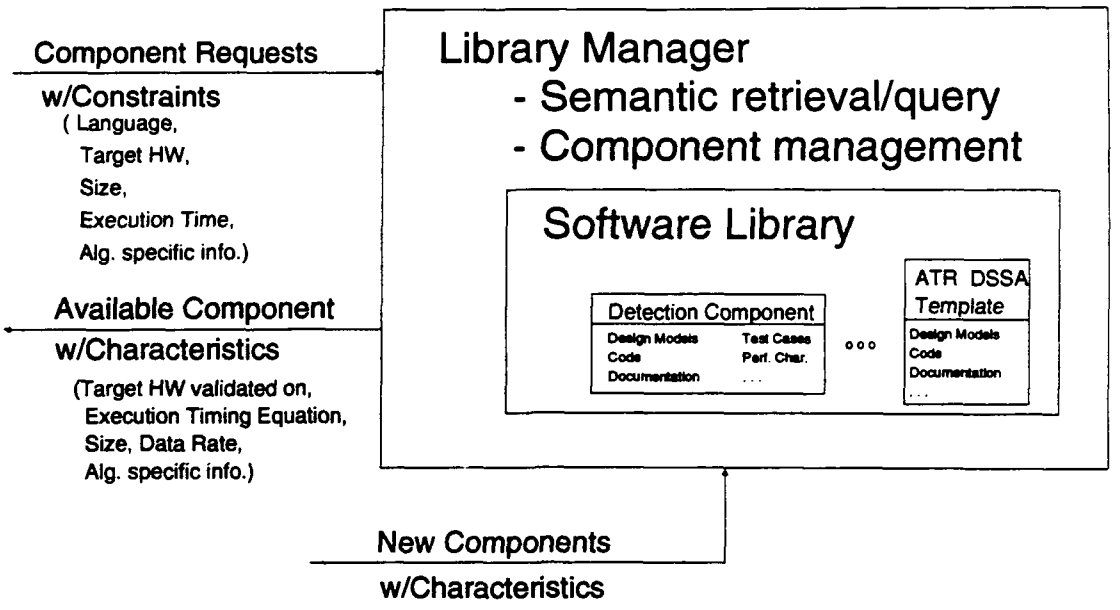


Figure 20
Software Design Overview

5.2 Software Library

Reusable software libraries offer the greatest contribution to reduced in-cycle software development/upgrade time. As illustrated in Figure 21, the software library supports the in-cycle system design activities



Reuse Enablers:

Domain Analysis
Object-Oriented Component Designs
Database to Support Retrieval and Storage
Standard OS Interfaces

Reuse Issues:

Reuse vs. Performance
Available Libraries
Domain Analysis
Logistics
Design for Reuse

Figure 21
Software Library

by providing semantic retrieval and query capability. This allows the system designer to utilize available component information in design trade-offs, system simulations, and hardware/software configuration.

The system designer can access the software library to request components required to support the system or upgrade being designed. Additionally, constraints can accompany the request to limit search to those library components that comply with some predetermined requirements (e.g., execution time limit or size limit). The response to the system designer can be the various design record information available for the requested component or an "intelligent negative response", indicating that the requested component is not available, but also giving information as to which constraint failed, possible alternatives, etc. For example, a request is made for an Image Screening Detection Algorithm that executes within 40 msec on a specific processor and is coded in Ada. An answer may be that the requested component does not exist, but that an implementation that meets all constraints except that it executes in 43 msec is available. This kind of intelligent retrieval/query capability will assist the system designer in making the required trade-offs, increasing the probability that the system being developed/upgraded will meet its requirements not only in terms of performance, but cost and schedule as well.

In addition to retrieval/query support, the library manager supports the addition, update, and deletion of components to the library. Components in the library can range in complexity from individual modules to

templates of particular domain specific software architectures. Component information is not limited to simply code, but may include simulation performance models, design models, test cases, etc.

Along with the potential benefits, reuse presents many challenges to software development organizations in the next several years. Key to enabling effective software reuse are: domain analysis to determine what to put in the library; data bases and intelligent access tools to support retrieval and storage; object-oriented component designs and standard OS interfaces to decouple components from specific system instantiations; and incorporation of reuse into defined system and software development processes. These areas are being addressed by both the DoD and industry in the form of the Draft Department of Defense Software Technology Strategy, programs such as DSSA and STARS, SEI activities, and industry process improvement and reuse strategies. For the RASSP demo program, the results from the ongoing work in reuse must be used to develop a reusable library for the selected domain.

A primary reuse issue for real-time systems will be performance. The tendency for real-time software developers is to implement code in the most efficient time/space manner possible because there is always more functionality that needs to be put in the system. Accepting a less efficient implementation even though it will save greatly in software development costs has not often been an option. For reuse to be successful for these kinds of systems, a reuse/performance tradeoff must be made in light of cost and schedule. A change in attitudes from "requiring the optimum solution" to "allowing a cost/schedule effective sub-optimal solution" must be made. Reuse of software between systems implies that a sub-optimal solution will be used. The solution is sub-optimal in the sense that processor throughput, bus throughput, memory utilization, or algorithmic performance may not be as good as for a single point approach. For a given system, it is possible that reused software may not require hardware modifications and may still meet the system's requirements. Or, reuse may require additional hardware, such as additional processors and/or memory, but the system costs may be reduced by reusing the software and having the additional hardware. For each system, it depends on the software development from scratch cost versus hardware recurring cost versus system development schedule. The point being that reuse requires a look at cost/schedule from the system point-of-view over the entire system life-cycle.

Additionally, the mechanical and business issues associated with reuse must be worked out. Issues include library search and component access, component validation, configuration control, and maintenance. Because of the additional verification required and additional thought in making sure the software is designed for reuse, reusable software is more expensive to develop than software designed for a point solution. But, if reused enough, the cost is amortized over multiple system developments and the end result is lower cost. Business issues to be considered relate to warranties, ownership, getting proprietary software into reusable libraries, and determining responsibility for and ensuring maintenance of reusable components.

5.3 Software Engineering Environment

Development of software components for a reusable library will require a combination of process, methodology and tools. The software development process is iterative, with feedback to preceding activities required. For example, results from a performance analysis simulation may affect the requirements analysis, and in turn the software design. The process must accommodate this iterative feedback and insure that affected activities respond accordingly. Key to improving the software process is the ability to assess it. Through the SEI's Capability Maturity

Model (CMM) for software, this assessment and process improvement is starting to take place. The SEI has identified five levels of software process maturity, shown in Figure 22. Currently, the majority of DoD contractors are executing software development at level 1 or 2. This low level is primarily due to the lack of a well-defined and documented process. The current process improvement work being done by the SEI and the process fanout underway in the software industry are contributing to the improved software development processes required for RASSP.

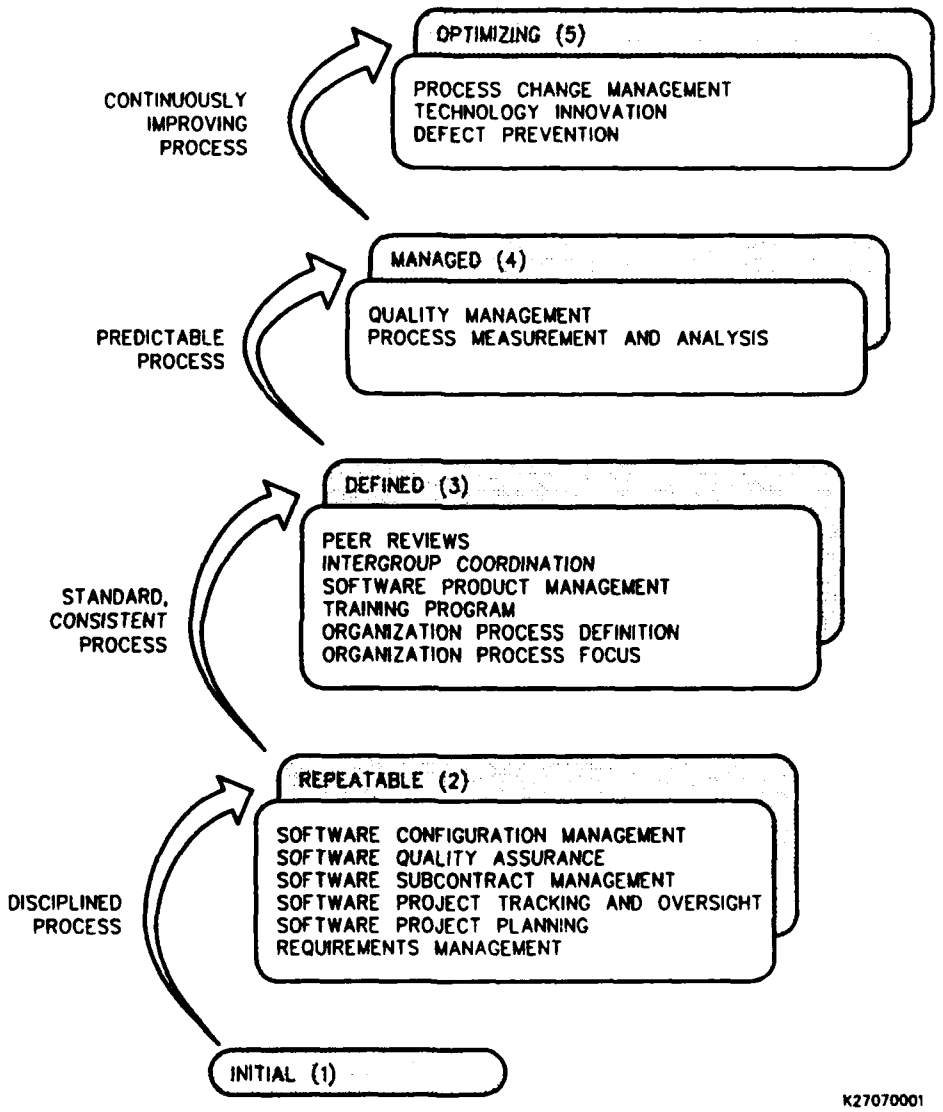


Figure 22
Five Levels of Software Process Maturity

A software methodology can be thought of as the "how to do" for the "what to do" specified by the software process. In actuality there are many parts of a software methodology – a requirements analysis methodology, a design methodology, a simulation methodology, a testing methodology, etc. They must all work together to support the overall software process. Currently, many discussions focus on the analysis and design methodology options. Structured Analysis/Design has grown popular over the last few years and is fairly mature. Relatively new in actual practice, but gaining popularity rapidly, is Object-Oriented

Analysis/Design. This approach seems to yield a better mapping of the problem domain to the solution domain. Additionally, since objects and the actions performed on the objects are encapsulated, the designs lend themselves more readily to reuse. As the methodology matures and stabilizes and tools better support it, the shift will likely be toward Object-Oriented Analysis/Design.

To effectively and efficiently develop the components to be included in the software library, there must be a comprehensive set of tools to support the development activities from requirements analysis through component testing. These tools should provide capabilities for:

- Requirements analysis and design
- Executable specification
- Requirements traceability
- Simulation to support partitioning/mapping analysis
- Reverse engineering
- Translation tools for code generation
- Host and target level debuggers
- Documentation generation
- Configuration management
- Metric collection and reporting
- Project planning and tracking
- Reuse library component retrieval.

Figure 23 shows a process flow for software development the way it is done today. The boxes represent activities. The entries under the boxes indicate commercially available tools that can be applied to that activity. The circles represent the consolidation of the design information that the activities both feed and draw upon. We can develop a software requirements model from which performance simulations, system test cases and documentation can be generated. The documentation and requirements model then feeds the implementation design. We develop another set of models for the implementation to which PDL and code can be added. From these annotated structure graphs, tools can be used to analyze complexity, generate test cases and analyze test coverage, produce more documentation and generate the source code that can be compiled and linked. Testing cycles from unit test through system integration are supported using host debuggers, target instruction level simulators and eventually the target system and target test system to finally produce a deliverable system. Underlying these development activities are the project management, configuration management and metric collection and reporting tasks. The tools to support the development activities are emerging in the commercial markets, but are relatively immature. Some activities are better supported by tools than others. Currently, some of the weaker areas are in requirements tracing/management, testing, and transition from requirements to design. Some vendors have chosen to specialize in a given area, while others have tried to cover the breadth of activities to some degree, whether integrating capabilities into a single tool's environment or providing interfaces for other vendor's tools.

For existing systems, tools are also becoming available that help support software reengineering via reverse engineering source code into annotated structure graphs. Once the structure graphs are obtained, the same activities previously discussed can be performed to reengineer and regenerate the software.

As we look to the future, what changes can be expected to be available in five years? For the most part, the development will be for components that go into a domain software library as opposed to specific system implementations. This does not mean that there will not be large components since a given component can represent some large software

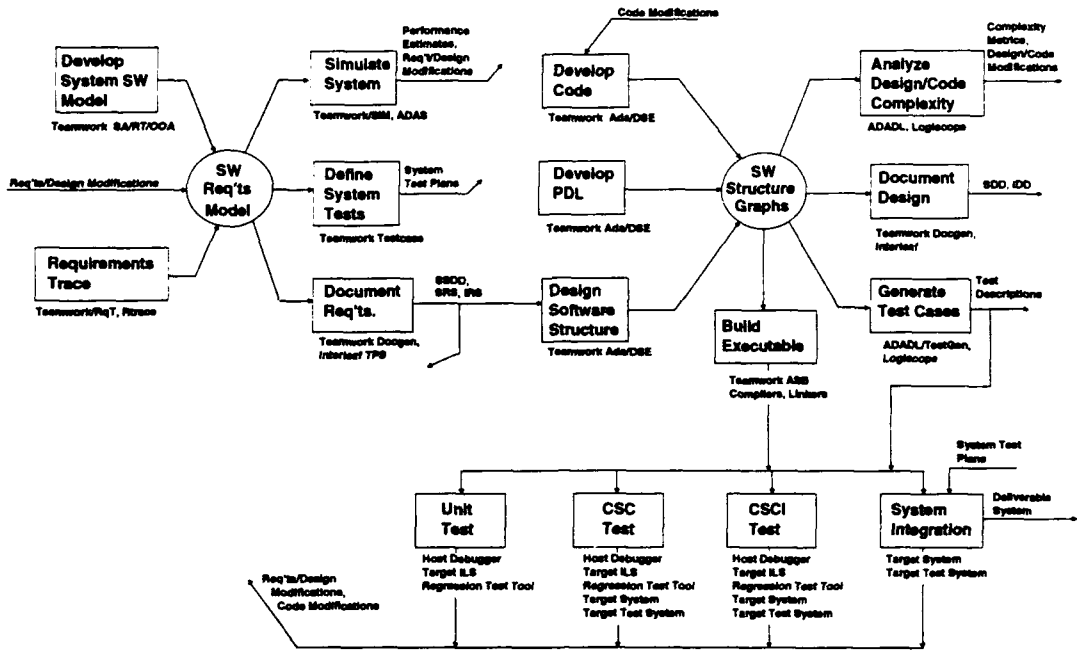


Figure 23
Software Development - Today

subsystem. It just means that the design for reuse tradeoffs will become an established part of developing software. In general, the tools that are emerging today will have increased capabilities and be better integrated with one another. As shown in Figure 24, reuse libraries will be available to draw upon for software component development just as the system designer draws upon hardware and software libraries at the system level. One area that will likely see increased automation is the transition from requirements to implementation design. As we shift from Structured Analysis/Design to Object-Oriented Analysis/Design, the requirements and implementation views should be more compatible and therefore the transition more easily automated. With the release of Ada 9x and results from ongoing parallelizing compiler and operating systems research, there should also be better support for multiprocessor implementations.

By the year 2002, software developers will be performing the same basic activities, as depicted in Figure 25, but the level of automation will be significantly increased. Instead of the developer being responsible for initiating a reusable component search, the tools making use of the reusable information will interact with the reuse tools to initiate the search and retrieve the information. The transition from a requirements model representation to a different implementation model representation will be eliminated. The ability to easily execute and simulate the component model at the initial requirements stage and throughout the development will help in verifying that the designs meet requirements. We will have automated support for test case generation, coverage

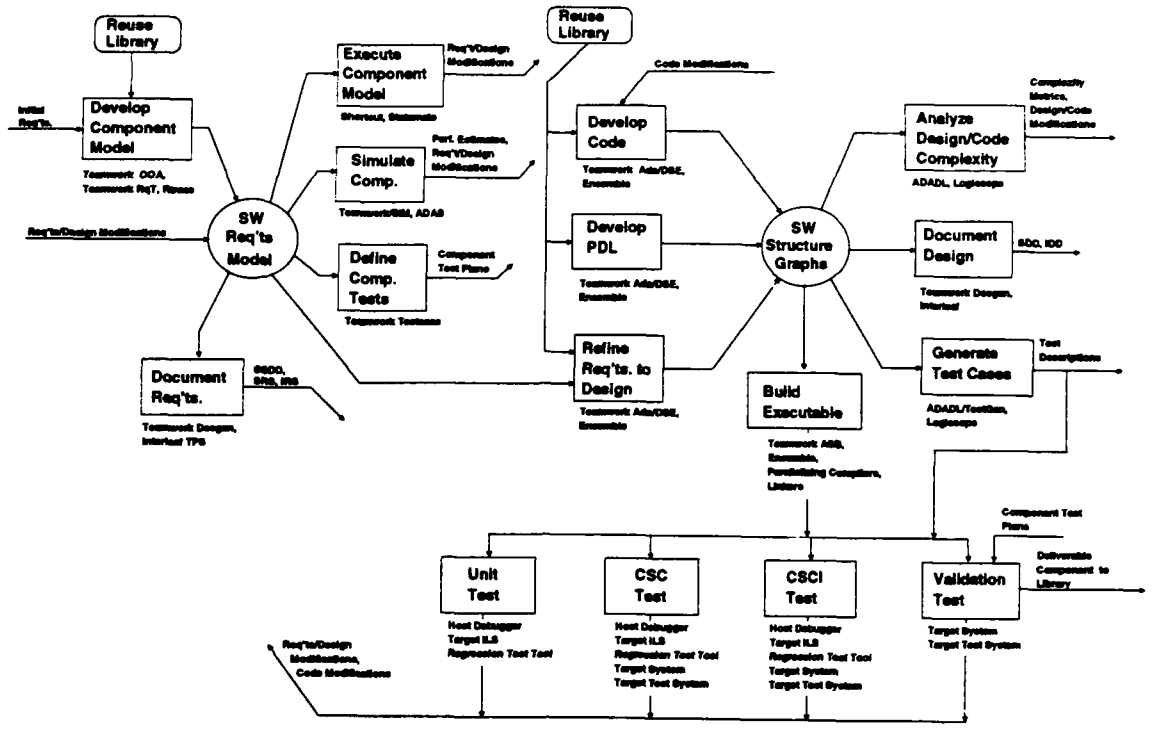


Figure 24
Software Development - 1997

analysis, and regression test management. Increased capabilities in automated code generators, parallelizing compilers and multiprocessor linkers will help place the emphasis of the development effort on the requirements analysis and capture, and the requirements and design verification via executable specifications.

The tools required for the software engineering environment proposed in the RASSP vision are emerging. Some examples of tools available to support the development activities are listed in Table 15. The tool vendors are, for the most part, committed to developing the tools required to automate the activities and providing easy information flow between tools. Business alliances are growing out of the realization of the need for this easy information flow and the recognition of various vendor's strengths and weaknesses. Vendors are involved in standards groups, working to define interface standards to support these needs. In the next few years, the tools will mature and gain an experienced user base. In the areas where tools are lacking capabilities or are nonexistent, DoD and industry in general, and RASSP in particular, are tasked to define the requirements and make the demand known to the commercial tool vendors.

5.4 Summary

The bottom line for software within the context of the RASSP vision is that much of the required effort has been identified, a DoD software strategy has been defined, and the appropriate areas are being worked by various government, academic, and industry efforts. For RASSP, we must maintain awareness of ongoing research, make sure efforts are coordinated, accelerate development when necessary, and feed back requirements to standards groups and commercial tool vendors. We must

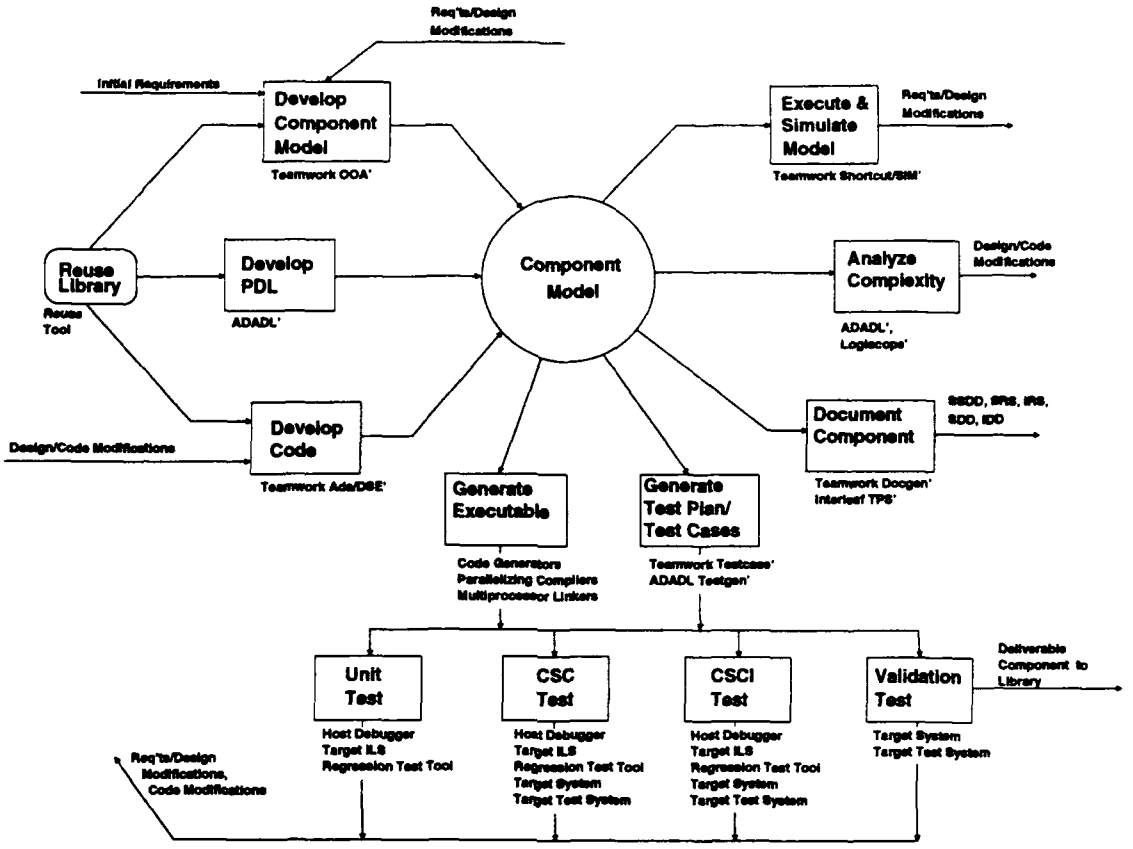


Figure 25
Software Development - 2002

use the results from these research and commercial efforts to develop components for a reusable library and establish a supportive software engineering environment.

Table 15 Software Engineering Environment Tools

Activity	Supported By	Comment
Requirements Analysis and Design	Teamwork	• Fairly well established, good overall environment approach
	Software Through Pictures (STP)	• Good capabilities, more niche oriented
Executable Specification	Statemate, Teamwork/ Shortcut	• Initial rapid-prototyping capabilities, need a standard formal specification language
Requirements Traceability	Teamwork/RqT, Rtrace, RDD-100	• Initial capability • Flexible, good potential • Powerful, but hard to use
Simulation	ADAS, Teamwork/SIM	• Mature, future questionable • Good initial capabilities, good integration with req. analysis and design • Need to work link to hardware • Good statistical simulation capability, limited integration with req. analysis and design
Reverse Engineering	Teamwork, STP	• Initial capabilities to generate Ada structure graphs/C structure charts from code
Translation Tools for Code Generation	Many vendors	• Primarily uniprocessor support currently
Host and Target Level Debuggers	Many vendors	• Primarily uniprocessor • Multiprocessor architectures require arch. specific target test systems
Documentation Generation	Interleaf, Teamwork, STP	• Good capability for extracting model information for documentation
Configuration Management	Caseware CM, Softool CCC	• Direction is to interface with Teamwork and STP type tools to provide CM
Metric Collection and Reporting		• Immature support, mostly manual, limited assistance available within tool environments
Project Planning and Tracking	Various vendors	• Not well integrated with development tool environments
Reuse Library Component Retrieval	Westinghouse Reuse	• Prototype reuse tool

6. COMPUTER AIDED DESIGN

6.1 RASSP Hardware CAE/CAD Investigation

Texas Instruments has investigated the critical aspects of hardware Computer Aided Engineering (CAE) and Computer Aided Design (CAD) as they relate to RASSP hardware design. The emphasis of this investigation has been on requirements and current status of design automation tools that support the library based design approach that Texas Instruments believes is a strategic process to the design and manufacture of RASSP modules and systems.

Design Automation is a key process for cost effective RASSP model year upgrades. For hardware design in a library based Application Specific Signal Processing environment, as shown in Figure 26, there are two key elements:

1. Design tools that support a range of analysis and synthesis capabilities in a top-down based concurrent engineering environment that addresses all critical aspects of design and libraries of signal processing subsystems and modules. These design tools may be used for processor design (in cycle mode) and for library model development (out of cycle mode). These tools include concurrent engineering and trade off analysis tools as well as electrical and mechanical capabilities.

2. Predefined libraries of signal processing subsystems, modules, and electronic components that support the range of design views and formats needed to effectively use the design tools in concurrent engineering design processes that allow emphasis to remain on the system and functional level aspects of the design. These libraries are developed in what is referred in the TI vision as the out of cycle mode, since the goal is to have fully characterized libraries to support a RASSP design cycle.

It is expected that the same classes of tools would be used in both in cycle and out of cycle design efforts, with the primary difference in use occurring in the level of abstraction of design being performed.

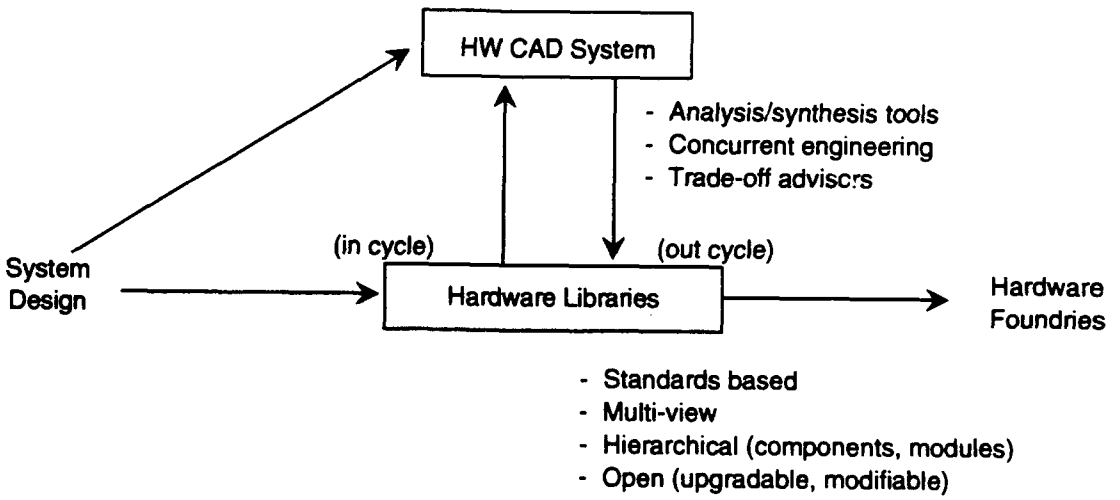


Figure 26
Library Based Hardware CAD Concept

A key concept in the RASSP HW CAD approach is to limit the amount of in cycle detailed design to the processor level design and to upgrade efforts that cannot be addressed outside the critical path. This concept, as shown in Figure 27, supports the development of a systems based design approach. Existing sets of design libraries provide a majority of the detailed design characterization and supporting model views.

Design work for the module elements is worked outside the critical path of the processor top-down design. For RASSP, this supports two key objectives;

1. The ability to design and analyze both original design and model year upgrades without having to manage underlying levels of design complexity.

2. A process for performing correct by construction synthesis and analysis of new library elements outside of the time constraints of the processor critical path. This should result in improved design and characterization of the module level designs.

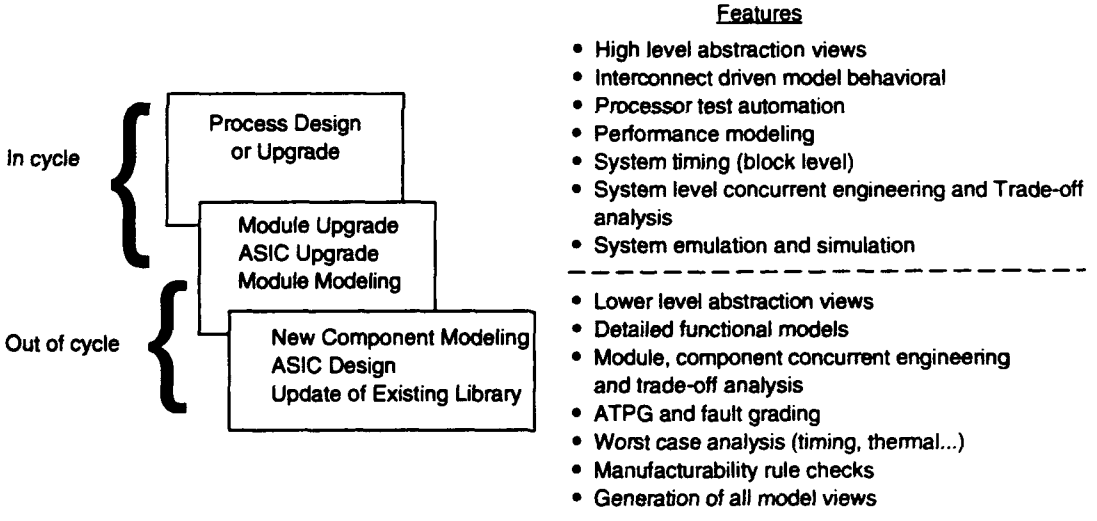


Figure 27
In Cycle and Out of Cycle RASSP CAD

Five classes of design automation tools can be identified as being critical and appropriate for both for development of RASSP library modules and processor designs.

Capture: Tools are used to represent and document design objects for further engineering efforts. Capture tools can be identified as graphical or textural. Graphical tools include block diagram and schematic level tools for electrical design. Mechanical graphical capture tools include mesh and 3-D element models. Textural capture

tools for electrical design include HDL modelers (i.e., VHDL, PALASM, AMD) for digital design, MAST (Analogy) and SPICE for analog design) and table based (i.e., truth tables and state tables for digital design). An aspect of overlap between design capture and the synthesis and translation steps is the automated generation of textural descriptions from block level graphical capture and generation of graphical representations of the design from textural descriptions. This flexibility in levels of interoperability between different capture mechanisms appears to be a key to providing an environment that allows engineers to use the capture mechanisms that they are most comfortable with.

One expected result confirmed during the investigation was that the level of textural based capture is much higher at the system and functional levels of design capture. This approach supports the concepts of textural HDL descriptions being used as "executable specifications" for design requirements and for HDL basing of both processor and library design activities on correct by construction synthesis approaches.

Analytic: Tools that support the functional and parametric analysis of designs are necessary for validation of design activities. Analytic tools exist in a number of design domains for both electrical (digital and analog) and mechanical disciplines. Examples of electrical analysis tools include simulation; both for behavioral and structural designs (VHDL and SPICE based tools), and worst case analysis tools (worst case timing analysis and Monte Carlo analysis). Examples of mechanical analysis tools include finite element tools for stress modeling and thermal analysis tools.

A key requirement for RASSP would be to have analytic tools capable of supporting the level of analysis required for system functional level design steps. While point solution tools to address system design are available from numerous vendors, there is limited standardization and integration of tools exist at this level.

Synthesis: Tools that support the transformation of design data from one representation to another based on analytic rules. Synthesis tools include both optimization and mapping (primarily in use with digital designs) of design data between design abstraction levels and interface tools that translate one design representation to another equivalent format. A differentiating point between synthesis mapping and translation is that while design information can be increased during synthesis mapping, due to the increased detail provided by moving between abstractions, translation is limited to a uniform information level and typically results in loss of design information during movement between formats.

A significant emphasis by CAD vendors has been on addressing synthesis for two areas of RASSP related design:

1. Logic synthesis and optimization of register level digital designs. This approach works well with control components of a digital design and has been widely commercialized by CAE vendors such as Synopsis and Mentor.

2. Algorithm to silicon synthesis supporting both generation of assembly code for standard DSP components and datapath elements of specific widely used signal processing algorithms (FIR, FFT, etc.). This approach has been addressed by both CAE and ASIC vendors.[1,2]

Areas of critical synthesis development where no existing commercial tools exist include synthesis of general purpose analog designs and

synthesis from processor level specification and performance level system representations Work is ongoing in both areas at the university level, which indicates a high probability of migration to the commercial environment in the next three to five years.

Tradeoff Analysis: Tools that support the tradeoff of parameters and provide recommendations and design advice on areas in the design space that can provide a feasible solution to diverse design requirements. A limitation on first pass success of a RASSP design is the ability to completely cover all model view and design variables in a given design phase. A key goal in a design or upgrade is to ensure that the design is centered with regard to all addressable parameters. This level of tradeoff is typically heuristic in nature, since dimensions and metrics for different parameters are not uniform or convertible and are typically application dependent.

Successful approaches in existing tradeoff analysis development are primarily university and research based and relied on heuristics from several design disciplines (concurrent engineering). Development has focused on tradeoff analysis rules for limited portions of design space for specialized applications. A chart of some applicable tradeoff analysis tools for RASSP is given in the Foundry CAD Section. Tradeoff analysis is typically limited to a single level of design abstraction and to specific signal processing types.[3] This implies a development of several classes of tradeoff analysis tools to support a RASSP design process, each addressing a specific level of design concerns at a given level of the top down design process. To address aspects of systems engineering of RASSP, the point solutions tradeoff analysis tools should ideally be able to communicate the statistical design information among themselves to provide optimal design centering of all the concurrent engineering portions of the design space.

A viewgraph presented during the Hardware CAD discussion at the RASSP final review, but not included in that viewgraph package, is included as Figure 28.

Concurrent Engineering: Tools that support the manufacturability and other "ility" processes included in the product life cycle. These tools vary over a range of design areas including testability, reliability, maintainability, producibility, repairability, human factors, etc. This class of tools typically used a mixture of analytic and rule based heuristics. In some disciplines (testability, reliability), well defined analytic approaches exist that can be applied to RASSP design. In other areas, concurrent engineering is very highly correlated to specific design and manufacturing processes. For these cases, the level of formalized analytic rules for concurrent engineering is typically much lower and more reliant on engineering heuristics and "rule of thumb".

There appears to be a high level of overlap between concurrent engineering and tradeoff analysis level tools. In many cases, integration of tradeoff analysis for different concurrent engineering disciplines appears as the key metric in determining the level of "quality" of the final design.

There has been limited success by commercial CAD vendors in integrating concurrent engineering into tool suites. This appears to derive from the multi-dimensional aspect of concurrent engineering, where integration of both electrical and mechanical design information must be integrated. In some cases, for example reliability, well described analytic and heuristic rules exist (MIL STD 217) but component level information for complex design is not typically available.

One domain of concurrent engineering, testability, has a significant level of impact to multiple levels of the design process that merits additional discussion. Testability is probably the most mature of the concurrent engineering disciplines, in part due to its primarily electrical information content and also due to the level of formalized and widely agreed upon analytic processes that support electrical test.

Testability is key in RASSP, both for design and upgrade verification and for life cycle maintenance. Developments in recent years have both matured the level of test CAD support available and increased the level of design and test interaction. The ability to embed testability in a design via synthesis and standardized test structures, and the capability to formally generate and measure test information via CAD tools such as Automated Test Pattern Generation and Fault Simulation, radically improve the testability of a design. Development of standardized test data formats (WAVES, EDIF test) complement existing capabilities such as ATLAS and design to manufacture test interfaces. An open area of development is standards for formalizing test information; which is initially being addressed by Boundary Scan Description Language (BSDL) for JTAG test standards.

Observations on testability analysis, as it applies to RASSP, indicate that additional development work is required in the key areas of fault simulation and ATPG. The ability to fault simulate at higher levels of design abstraction (i.e., HDL) is not being adequately addressed in industry. Automated Test Pattern Generation techniques currently in use are driven by providing a level of vectors to meet fault grading requirements, they often cannot provide the level of discrimination to address aspects of analysis that are of interest to the designer for a given analysis. Both these areas appear to require additional

- Model year upgrade concepts imply the need for domain specific trade-off
  - Design size/complexity bottleneck in ASSP upgrade trade-offs
  - Variable and parameter size increases with each upgrade
  - Design traceability to specification without regression testing

- Design advisers and expert systems are leading edge productivity multipliers
  - Tools should address multi discipline concurrent engineering
  - Statistical "design centering" of each upgrade

- Trade-off analysis needed at several step in design flow
  - Expertise in defined domains and abstraction levels
  - Object oriented data bases to provide unification
  - MCC design advisor (MSDA) prototype as an example

- Recommendations:

  Expert systems based tool development acceleration and prototyping

Figure 28
RASSP Design Advisers/Tradeoff Analysis

conceptual research to meet the above requirements. An overview of current trend testability analysis areas as they apply to RASSP design are given in Figure 29.

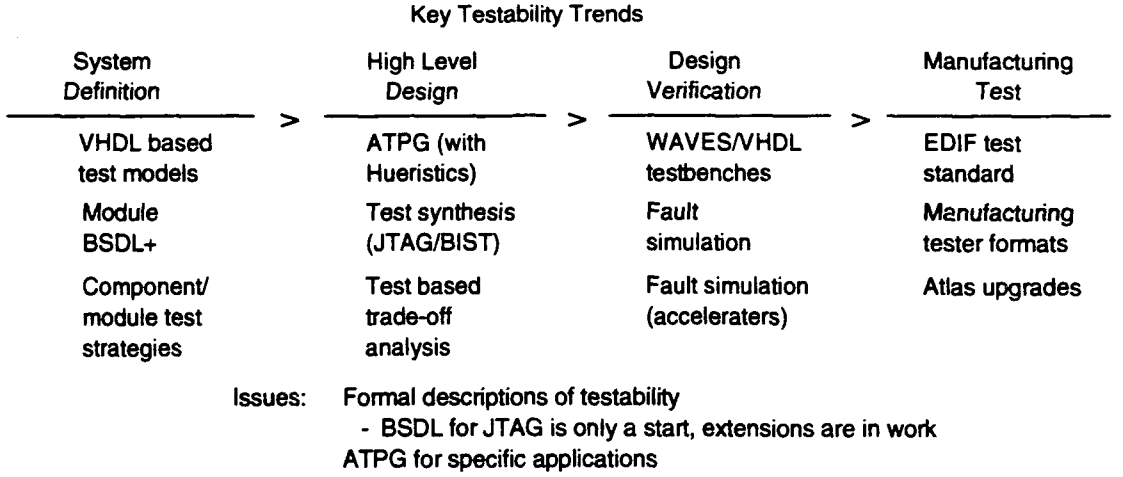


Figure 29
Key RASSP Related Testability Trends

Much of the RASSP relevant analysis and synthesis capability needed in this design environment is being actively addressed by mainstream CAD vendors. Vendors appear to be concentrating on improvement of core analytic based tools and expanding their focus based on evolution of hardware design capabilities. Recent development trends have focused in interoperability, based on standards support and framework based compatibility. Key RASSP design needs such as heuristic based tools, applied tradeoff analysis, and system modeling at the conceptual level have been being inadequately addressed, although some development trends indicate an increased interest in these areas.

Some CAD vendors involved in detailed discussions during this investigation and who are developing products suites that apply directly to the RASSP effort are:

- ANALOGY. Analogy produces behavioral analog design tools (SABER) including behavioral libraries for RASSP elements such as DSP, sensors and controllers. Analogy has demonstrated mixed mode simulation capabilities with multiple third party vendors. Analogy is working on new products to address statistical design and failure mode analysis.

- COMDISCO. Comdisco focuses on system and DSP design tools (SPW, BOSS, BONES, etc.) with support for VHDL, C, and ADA based designs. Comdisco supports a large and mature suite of DSP model libraries and has demonstrated design tools for parallel DSP processing and interfaces to third party synthesis capabilities.

- MENTOR. Mentor produces a diverse and well established tool suite based on an object oriented database and design framework. Mentor allows interfaces with third party design automation tools. Mentor has released a "DSP Station" product that supports algorithm to silicon design paths from descriptions in the Silage DSP data flow language. New products that would support RASSP design include "System Station", a tool suite to support requirements capture, generation, and analysis.

6.2 CAD Process Flow Overview

Top-down design has been generally accepted as an effective means of managing design complexity. Typical top-down flows used at Texas Instruments for both electrical and mechanical design (along with vendor and internal tools that support these flows) are given in Figure 30a and 30b. A merged library based top-down design flow that would support RASSP design modifies this design flow by moving the detailed analysis and verification steps out of the critical path. A view of this flow showing the extraction of detailed design and verification is given in Figure 31. For RASSP, this modification supports two key objectives: the ability to analyze the model year upgrade without having to manage underlying levels of design complexity; and a process for performing correct by construction synthesis of new library elements once the model year upgrade design has been completed. As in the traditional flow in Figure 30, initial design and upgrades can enter the design flow at any point and be able to leverage the advantages of top down design. The following sections will address individual levels of the top down CAD flow in more detail.

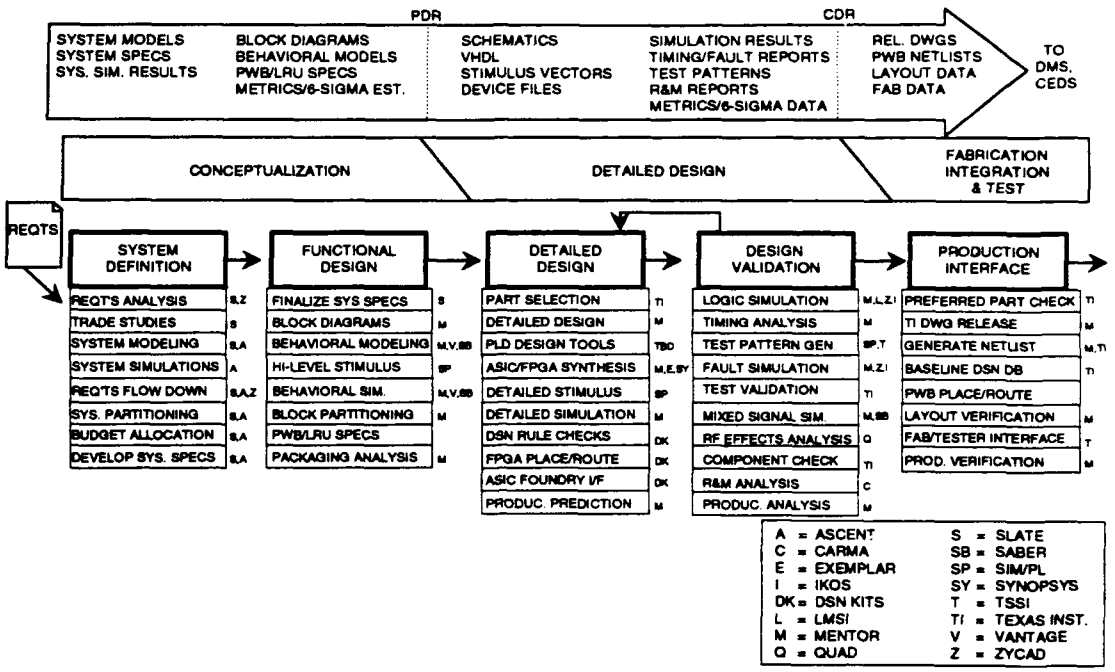
6.2.1 System/Concept Level

Due to the level of abstraction, system and conceptual design presents the greatest challenge from a design automation viewpoint. The design tasks are typically diverse and heuristic, with few standardized formats for design description or performance. Current efforts to address the formalization of system requirements are at an immature stage. This is a limiting factor on both use of design tools and their integration. VHDL is an output format in some classes of these tools but is not considered sufficient to address RASSP design and tradeoff complexities. PDES and other parametric standards can be expected to support aspects of design formalization and tradeoff analysis but do not provide a comprehensive system level design view. Typical design tasks and input and output format for this level of design are given in Figure 32.

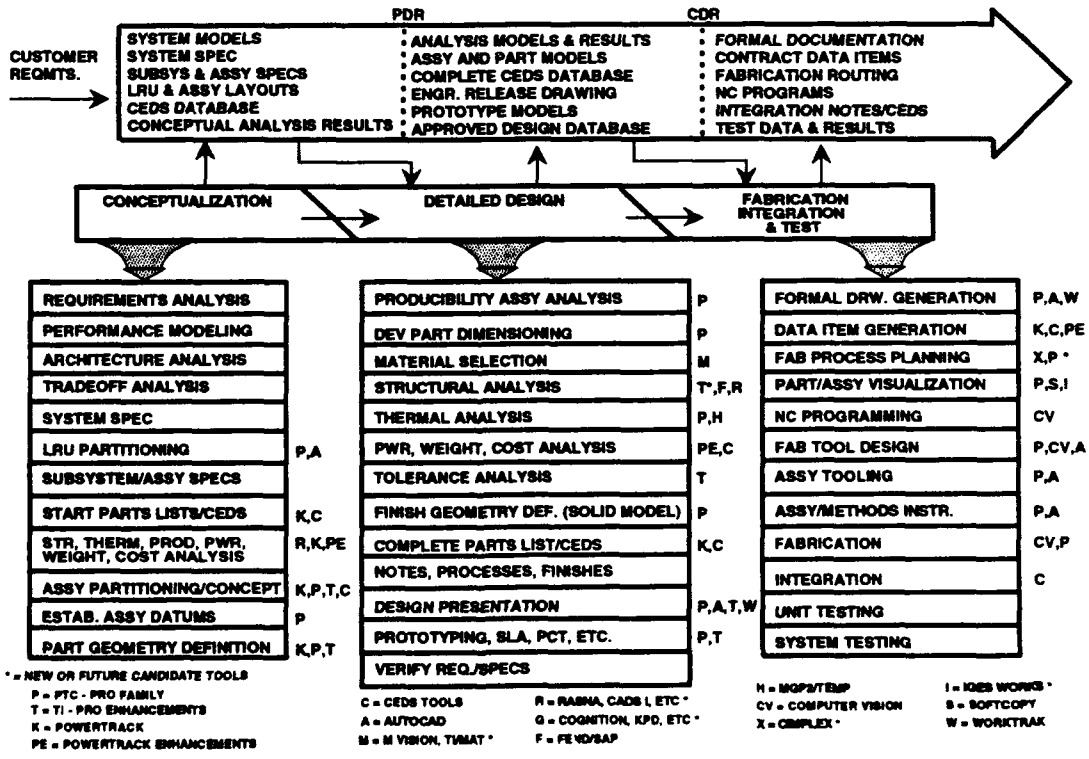
6.2.2 Functional Design Level

The functional phase is the point of design realization where a majority of the hardware design decisions and tradeoffs would be made in both initial and model year upgrade efforts. In an idealized 10 development window, most, if not all other the lower level design efforts could be synthesized from the functional model view. In current design efforts, the functional view is where design model standards begin to be effectively utilized, and where the lack of model standards in areas becomes a critical limiting factor. Model views and tools need to support the decision process in selecting components and module partitions to meet performance goals using incomplete or statistically derived information.

Since the functional level of design is typically the lowest level of detail that a process designer would be addressing, the ability to reuse model views for different types of analysis is important for design consistency. For digital portions of a RASSP design, VHDL is expected to be used extensively for behavioral and register level models. Typical design tasks and input and output formats for the functional level of design are given in Figure 33. Most of the example tools presented at this level are VHDL based or interoperable with VHDL models.



(a) TI ECAE Design Flow



(b) TI MCAE Design Flow

Figure 30
TI Design Flow

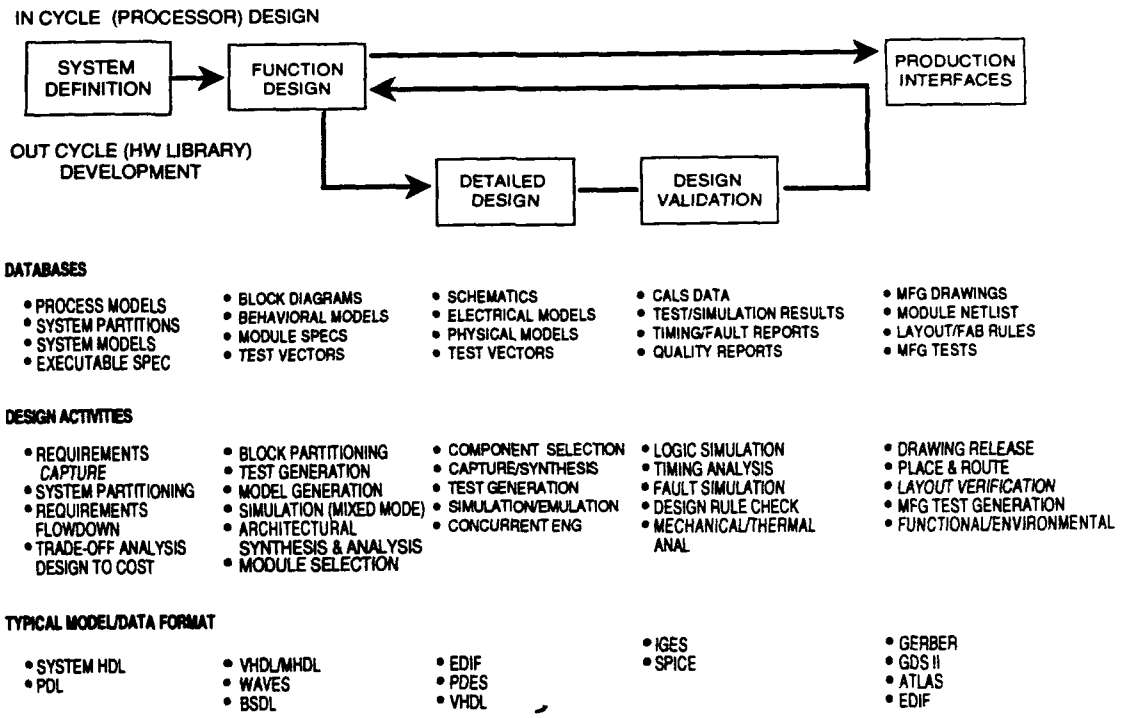


Figure 31
RASSP Design Flow

System Definition/Concept Design

CAD Inputs

- Requirement data
- Design constraints (speed, weight, space...)
- Design goals (throughput, power, heat...)

CAD Outputs

- Performance models
- Executable specs
- System partitions
- Performance data

Key Design Tasks

- Requirements capture
- Statistical capture
- Statistical analysis
- Performance simulation
- Spec level model gen
- System partitioning
- Performance trade-off

Example Tools

- SLATE (TI)
- MATLAB
- ADAS (CADRE)
- System station (Mentor)
- IDAS (JRS)
- ADAS

Issues: Need for S (System) HDL for performance are VHDL, PDES compatible with SHDL model needs for tradeoff anaysis tools for budget optimization partitioning centering performance trade-off HW/SW requirements.

Figure 32
System Definition/Concept Design

CAD Inputs

- Performance models
- Executable specs
- System partitions
- Performance data
- Test goals

CAD Output

- Parts selection
- Timing estimates
- Data flow simulations
- Block diagrams

Key Design Tasks

- Bus/behavioral modeling
- Design for test
- I.O. partitioning
- HW/SW trade-offs
- M/P/S partitioning
- Producibility modeling
- Mixed more simulation
- DSP algorithm development
- Performance trade-off modeling

Example Tools

- Vantage VHDL
- Expertest
- I-Logic
- IDAS
- ADAS
- Calce (U. of Maryland)
- Saber (Analogy)
- COMDISCO
- PDL (U. of Cincinnati)

Issues:
- Mixed more simulation, analog modeling standards, model availability
- Parallel processing analysis support is limited
- Performance/function/test description models
- Trade-off tools needed for logic/functional partitioning, module Performance, module producibility

Figure 33
Functional Design Level

6.2.3 Detailed Design Level

One of the key differences between current top-down design flows and the proposed RASSP flow is the removal of the detailed design level from the critical path and refining it into an out-of-cycle library development effort. From a design flow viewpoint, the detailed design stage can be considered a refinement of functional design. The flow from functional to detailed design is notable for the introduction of two design elements: analysis of design elements based on functionally complete and detailed model views that exist in a RASSP library; and the introduction of concurrent engineering tools that address the life cycle prediction and tradeoff of the design or upgrade. Model views at the detailed design level would typically correspond to "off-the-shelf" parts at a module or (standard or ASIC) component level. In current design environment, an increasing number of synthesis options exist to support handoff to the remainder of the design flow.

The increasing role of synthesis as a predominant detailed design approach in a ten year RASSP development timeframe is driven by the need for correct by construction design of increasingly complex module level components. The increasing use of synthesis is supported, in part, by

the availability and applicability of standard design formats for description of detailed design requirements and processes. Increased definition and flexibility in the model views at this level of detail should act as a driver for improved, more robust types of synthesis tools for detailed design. An emerging class of analysis tools for the formal verification of designs should also benefit from standardization and expansion of description formats.[4] Formal verification, previously applied to software design, has developed into classes of verification tools that operate from VHDL descriptions. Formal verification appears to a promising means of validating functionality in moving modeling and data views between in cycle design and out of cycle library development.

Typical design tasks and input and output formats for detailed design are given in Figure 34.

6.2.4 Design Verification Level

Verification of a RASSP design is identified as an independent task from detailed design since it addresses two key objectives:

1. The verification of the detailed design against the conceptual design specifications and requirements for the library component.

2. Analysis of a completed design element or upgrade for migration of design changes into concurrent engineering areas not anticipated during the design phases.

Although verification and detailed design are typically iterating in intermingled processes during the development of a library component, the goals at each step are seen as being distinct. Since the performance of a library based RASSP design is based on the credibility of the core library views, detailed and formal verification steps are critical. The time investment that must be made in validating the library views is one of the key rationals for having detailed design and verification as out of cycle processes.

The design verification phase is a key data generation stage in RASSP design since much of the detailed parametric and functional data needed, both for modeling views and building of new library elements, is created. This data includes the extended data items and data base deliverables, such as gate level representations and test vector files, needed for manufacture and test of the RASSP processor level product. Typical design tasks and input and output formats for the level of design verification are given in Figure 35.

6.2.5 Manufacturing "Pre Fab" Interface Level

The production interface is primarily oriented towards hand-off of data for RASSP manufacturing steps. Model back-annotation between geometric and mechanical CAD and the model verification stage is critical to refining design data accuracy and optimizing performance. The pre-fab data interchange is currently addressed by a series of informal standard formats for part and connectivity modeling. A key concern is formalizing the hand-off of rules and assumptions made during the design stages to the manufacturing foundry environment, which in a RASSP scenario may not be co-located with the design environment. Interfaces to manufacturing are addressed in greater detail from a manufacturing point of view in other sections of this report. Typical design tasks and input and output formats for this design to manufacturing handoff are given in Figure 36.

CAD Inputs

- Parts selection
- Timing estimates
- Data flow simulations
- Block Diagrams
- Test Pattern

CAD Output

- Schematics
- Synthesized logic
- Synthesized test vectors
- Simulations/emulations
- Timing models

Key Design Tasks

- ASIC schematic capture
- Synthesis modeling
- Stimulus generation
- HW/inst emulation
- Critical path analysis
- Component concurrent eng
- Packaging analysis
- Test insertion/synthesis

Example Tools

- Design architect (Mentor)
- Vista language assistance
- WaveMaker (TSSI)
- Crosscheck
- QuickPath (Mentor)
- CARMA (TI)
- Pro ENGINEER
- Synopsis

Issues:

- Synthesis of analog circuits
- Intelligent ATPG
- Testability, reliability, maintainability descriptions
- Trade-off analysis for final component selection
- Electrical/mechanical interaction, performance optimization,
- Module floor planning, placement, multi-ASIC logic partitioning

Figure 34
Detailed Design Level

6.3 Design Information Modeling

The different types of design information that would be used in RASSP model and processor design have been investigated. Information was segmented in terms of the types of data required for different design tasks. A rough grouping of data was determined to fit into three categories:

1. Functional data including algorithm and behavioral descriptions, logic and circuit representations, transfer functions, HDL models, etc.
2. Parametric and physical data including module and component electrical and mechanical data book information and graphical design data.
3. Performance data including the design specifications, goals, and requirements at different levels of design complexity and abstraction.

The relationship of these types of design information to existing data formats and tool types is given in Figure 37. For functional and parametric data types, several standardized or widely used data formats exist and are supported by major vendors. For performance data, there is no apparent data format that supports even the partial ability to

CAD Inputs
- Schematics
- Synthesized logic
- Synthesized test vectors
- Simulations/emulations
- Timing models

CAD Outputs
- Releaseable drawings
- Programmed parts
- Fault grades
- Sensitivity analysis
- Worst case timing analysis

Key Design Tasks
- Design rule checks
- Thermal analysis
- Test fault simulation
- Worst case tolerancing
- Part progamming
- Worst case timing analysis

Example Tools
- Check Mate (TI)
- Pacific Numerics
- Intelligen (RACAL)
- MDA (TI)
- DATA I/O

Issues: VHDL based worst base timing
Enhanced "ility" tools
Design verification to concept spec
Mixed mode acceleration
Packaging partitioning (interconnect driven)
Trade-off analysis for module tolerancing, sensitivity minimization,
Back annotation estimates, packaging partitioning (interconnect driven)

Figure 35
Design Verification Level

CAD Inputs
- Releaseable drawings
- Programmed parts
- Fault grades
- Sensitivity analysis
- Thermal models

CAD Output
- Module place and route
- Mfg test patterns
- Parts/net lists
- Module drawings
- Mfg. "ility" rules

Key Design Tasks
- Mfg drawing gen
- Module interconnect
- Testor input gen
- Module back annotation
- Part list/netlist

Example Tools
- AutoCAD
- MCMStation (Mentor)
- TSSI
- Quad design

Issues: Formats for design to manufacturing handoff not standards
Component foundry < -- > module foundry communication
Trade-off tools for mfg process selection, layout/fan rules,
"ility" rules

Figure 36
Manufacturing "Pre Fab" Interface

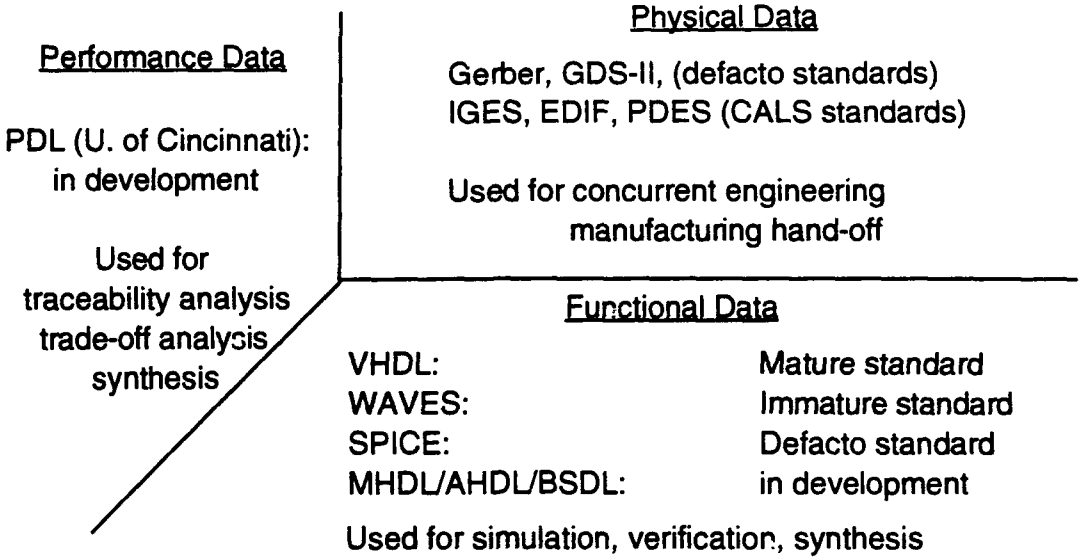


Figure 37
RASSP Design Information Modeling

model performance data. One effort that may lead to a partial solution is the development of a Performance Description Language (PDL) by the University of Cincinnati. This work is currently at a research level of development.[5]

Key to the RASSP concept of model year upgrade automation is efficient access of existing design information. A recurring theme addressed by diverse parties during the investigation effort is the importance of reusable, design specific, and mutually compatible model views of RASSP information. The information that potentially can be used in RASSP design and analysis can be very diverse. Some of the model views are shown in Figure 38.

Standardization of modeling formats and information is key to developing a comprehensive RASSP model library. Existing efforts have concentrated on functional and parametric data description standards. For top down RASSP design and model year upgrades, formalizing descriptions of design performance is equally important. A mapping of different standardized and widely used data formats and their applicability to different model views and design tasks is given in Table 16.

Standardization efforts that address these types of data and efforts to ensure harmonization of different model views are areas that need to be addressed in a long term RASSP environment. Current estimates are that this level of data harmonization has a ten year development horizon to commercial support, based on the number and data content of different data representation standards. Acceleration of this harmonization would have direct benefits to the RASSP effort.

6.4 RASSP Libraries

TI's vision of RASSP design is centered on the concept of reusable design libraries. From a hardware design viewpoint, a key requirement is to have module views that address a concurrent engineering process and support a top-down design flow. Model standards for several parts of this flow are immature. A key CAD trend in industry has been the

Table 16. Relevant Standards Versus Needs

	VHDL	EDIF	GERBER	GDS-II	WAVES	BSDL	MHDL	PDL	SPICE	PDES	IGES	ATLAS
BEHAVIORAL	XXX						XX					
STRUCTURE	XXX	XX					?	X	X			
PHYSICAL		X	XXX					X	X			
PERF. ATTRS.	XX							XXX				
CONSTRAINTS	X							XXX				
TEST STRUCTURES	X			XX	XXX	XX						X
TEST BENCHES	X				XXX							XX
DEVICE TECH.								X	XXX	?		
DEVICE MFG.										?		
PKG TECH.				X						?	XX	
PKG MFG.			XX	X						?		

- No Standard addresses the range of RASSP model needs
- Harmonized co-application will address short term requirements

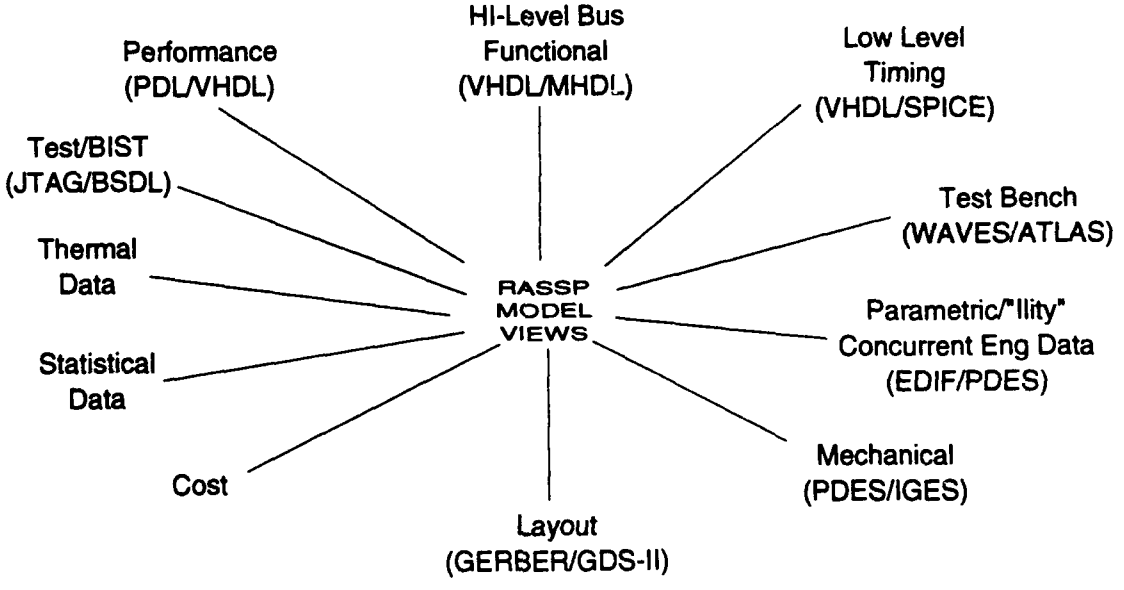


Figure 38
RASSP Design Data Views

migration to third party library vendors, the most notable from a digital design viewpoint being Logic Modeling Corporations (previously LAI). There are strategic problems with relying on this third part library approach, most notably the lack of control this implies in a library based process and in the level and scope of model views that third party vendors will support. An open library based on standards, where model views can be updated and optimized for needed level of analysis, is considered key to support of a RASSP design process.

At a component level, semiconductor vendors are beginning to make open models of their designs available to users. Several analog silicon vendors provide SPICE models of their designs. Open libraries of ASIC components are becoming available from several vendors. VHDL models are beginning to become available from some vendors. (TI has recently released a VHDL model for 16 Meg DRAM), but this is still not the norm.

A more detailed discussion of hardware library types and issues as they relate to the RASSP environment is addressed in other sections of this document.

6.5 Tool Data Base Federation

The RASSP design environment will typically be made up of multiple data base layers, containing data views in both object oriented and relational formats. These data bases may be distributed over multiple locations and interfaced via varying CAD frameworks. There are several areas of data base standardization and development that need to be supported in a RASSP design environment, including application procedural interfaces, standardization of data base structures and management, federated interoperability of the different data bases, and standardization of the varying levels of database access. These issues are being addressed from a CAD viewpoint by industry organizations, most notably being the CAD Framework Initiative (CFI), which plans to have initial protocol and data management procedures identified in 1993.

Increasing use of object oriented data bases appears to best support a RASSP multiple model view concept. OODB management standardization is being addressed by industry groups. A more detailed discussion of data base issues as they relate to the RASSP environment is addressed in other sections of this document.

6.6 Design Model Standards

RASSP requires widely accepted data exchange standards to ensure multi-partner design data interoperability and enhanced multi-vendor analysis capabilities.

Among the basic questions that exist for a RASSP CAD process are: (1) What are the minimal set of design model views needed to adequately represent a design and (2) What are the mechanisms and protocols to address data bases that contain RASSP libraries and design data. CALS (Computer-Aided Acquisition and Logistics Support) is the leading U.S. initiative that appears to provide a standard to address these and related issues. TI's RASSP vision plans to leverage and support CALS as a baseline for addressing data formatting and handling. Previously stated CALS goals can be summarized as follows:

- Integration of concurrent engineering into contractor CAD/CAE systems
- Replace paper deliverables with digital data
- Provide access to contractor integrated data bases

- Provide capability to receive, store, distribute, and use technical data in electronic form

source: 1988 CALS Report to Congress

RASSP relevant elements of the CALS approach include:

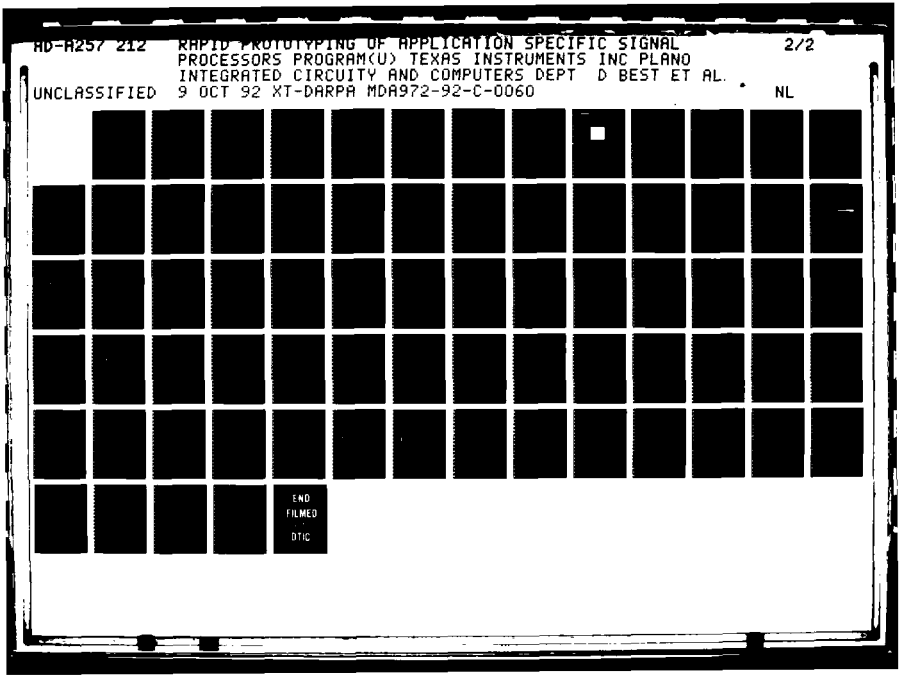
VHDL: VHSIC Hardware Description Language (VHDL) is considered to be one of the key standard data formats for RASSP design representations based on the proliferation of VHDL based tools and VHDL's role as the preferred design and synthesis language in digital design. VHDL can capture and model digital designs at various levels of abstraction and conveniently represent both behavioral specification and register level and structural design. This versatility gives VHDL considerable advantage over competing HDLs, such as Verilog, in the design verification process, this process can include simulation of functionally equivalent but abstractional different design representations. VHDL can model concurrently executing processes, which is essential for descriptions of both macroparallelism and microparallelism of digital hardware. Robust subsets of VHDL are supported as inputs formats for logic synthesis, and for test related data standards (WAVES, BSDL).

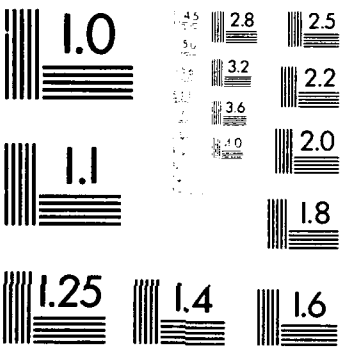
Since a majority of RASSP hardware design and upgrade is expected to be digital, VHDL plays a key role as a modeling language for RASSP. While there are obvious limitations to VHDL's role in system design, investigation and analysis indicates that VHDL can address the levels and kinds of model abstraction views needed to address digital signal processing design.

Several DSP specific data flow languages exist, a notable example being SILAGE from Cal-Berkeley. SILAGE is being used a part of the Mentor DSP Station tool suite.[6] As part of the investigation into applicability of VHDL to DSP design, several mappings were attempted between common SILAGE constructs and VHDL. In most cases, direct language remapping was identified, with the generation of some functions to support Z-transform and other DSP constructs. These functions were typically extracted from DSP support packages developed by Comdisco and Intermetrics.[7,8] The ability to generally mapping VHDL to DSP level functions is additionally supported by the availability of VHDL models as a design output from Comdisco's Signal Processing Workstation tool suite. Comdisco has demonstrated compatibility with VHDL subsets for synthesis from several vendors, including synthesis market leader, Synopsis.

The extension of VHDL to address analog and other continuous timing model views is a strategic limitation in support of system and architecture level RASSP design processes. The support of analog extensions to VHDL is being addressed on several fronts, most notably in the IEEE 1076.1 extensions being addressed by IEEE DASS committees. Additional work on extending VHDL is being addressed at the university level, notably by University of Cincinnati in their anaVHDL development. [9] AnaVHDL embeds the analog simulation environment in a VHDL program to allow a single-simulator simulation environment that supports both SPICE and VHDL models. Conversion functions are being developed as well. An outlined potential mixed mode environment using anaVHDL is given in Figure 39.

IGES: Intermediate Graphics Exchange Specification (IGES) is widely used by mechanical CAD vendors as a graphical data exchange format. Current IGES (Class II) operates under "lowest common denominator" theory of data exchange per MIL-D-28000. This is useful in that most systems can read the majority of simpler entities. It is harmful since it causes the loss of potentially useful information and increases the amount of


AD-A257 212
RAPID PROTOTYPING OF APPLICATION SPECIFIC SIGNAL
PROCESSORS PROGRAM(U) TEXAS INSTRUMENTS INC PLANO
INTEGRATED CIRCUITY AND COMPUTERS DEPT D BEST ET AL.
2/2
UNCLASSIFIED 9 OCT 92 XT-DARPA MDA972-92-C-0060
NL
END
FILMED
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

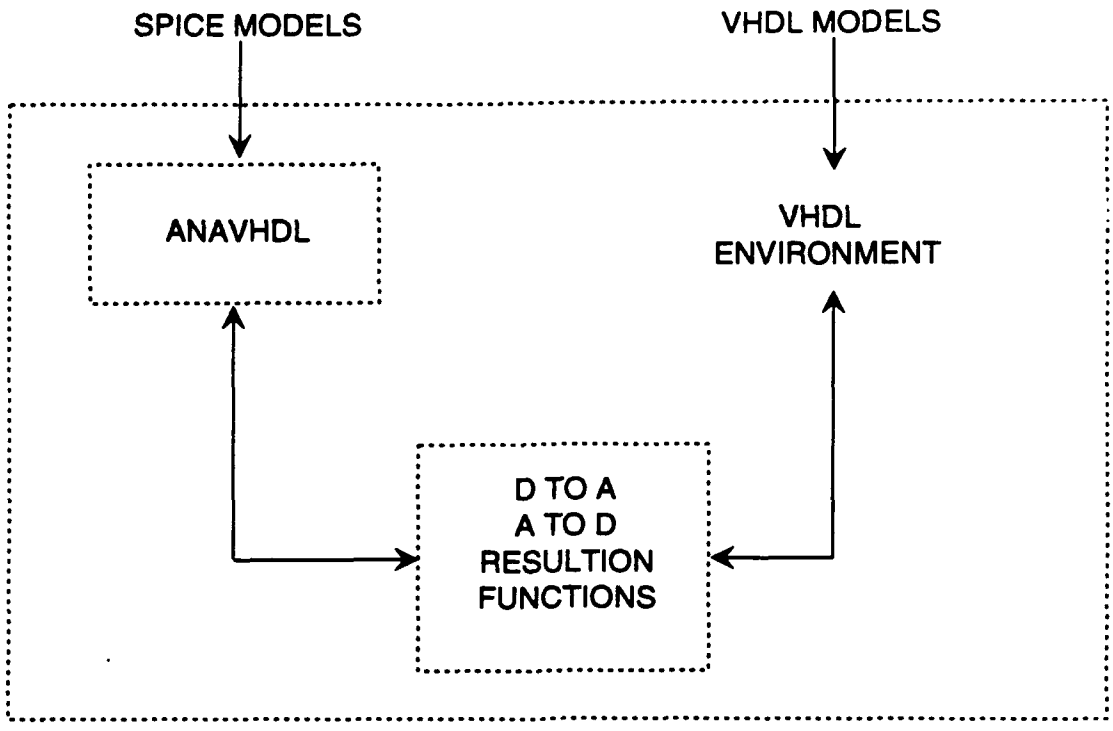


Figure 39
AnaVHDL Mixed Mode VHDL Environment

data to be handled. Current trends in IGES use has shifted to the development and use of application protocols that allow more complex predefined entities. This effort is still immature and levels and types of IGES application protocols that would benefit RASSP are undetermined. The DoD CALS office has halted IGES related funding in 1992. This will negatively impact the development efforts for more complex IGES information management.

PDES: Product Data Exchange using STEP (PDES) is an emerging design standard for parametric and physical information based on the joint Standard for the Exchange of Product (STEP) formats defined by U.S. and European development groups. One of the keys to CALS success as a RASSP interchange standard will be the acceptance of PDES as a design data standard. PDES is currently at a low level of maturity with initial STEP application protocols for electrical design not scheduled until 1993. One of the initial implementations of PDES will be under the Intermetrics PDES Application Protocols -Electronics (PAP-E) development effort. General consensus from the IGES/PDES Organization (IPO) is that PDES deliverables remain three to four years away.

How PDES ultimately addresses electrical design and parametric data and its interpretability with mechanical data, and whether it can comprehend the levels and types of design views that are critical for signal processing designs are areas that RASSP and RASSP participants can critically impact. Electrical/mechanical data interoperability identified as being critical for integrated modeling and analysis of RASSP designs is an area that PDES is expected to address. Previous addressing of electrical/mechanical data modeling under IGES as its Class III data format has resulted with virtually non-existent support from commercial vendors.

A key open issue for CALS in general, and PDES in particular, is how it will manage migration paths for existing design and legacy data. As a new standard, PDES does not have any identified mechanisms for upgrade paths from other data formats currently identified. This class of problem appears to be addressable by applied cross mapping of formats via application protocols. This has not been validated.

One of the key PDES initiatives has been the adoption of existing development efforts in other organizations, specifically, the adoption of the new generation of EDIF model views. The EDIF technical committees have re-structured future versions of EDIF significantly from the current essentially single view (EDIF 2.0) format. EDIF 2.9, which like PDES, is based on the Express Modeling Language supports 18 model views of the design and design process attributes. The supported model views for EDIF 2.9 are given in Figure 40. The roadmap for EDIF development over the next five-years includes test pattern and board level and MCM layout information (EDIF 4.00) as well as links to VHDL.

6.7 RASSP CAD Recommendations and Summary

Detailed design and verification level tools are evolving based on CAD vendor development. Architectural and concurrent engineering tools are being introduced from a number of sources and should become more available in the next three to five year timeframe.

There are several areas of hardware CAD infrastructure that need increased work for an optimal design environment for RASSP. These include tools for integrated systems engineering design. Systems engineering encompasses tasks in requirements capture, statistical analysis, performance simulation and tradeoff, and system level model generation. Current commercial vendor offerings do not appear to meet the range of RASSP requirements for systems engineering and design.

Concurrent engineering and tradeoff analysis is needed throughout the design process. This is perceived to be driven by concepts of design "centering". Design centering insures that optimal tolerance margins exist for all views of the RASSP product with regard to the concurrent engineering design, manufacturing and life cycle requirements.

Work needs to be supported under RASSP to address system design and tradeoff analysis/advisor class of tools. Additional focus on key areas such as formal verification and statistical modeling of designs is needed to provide robust tools for production use.

Design representation standards harmonization is key to a RASSP design process as defined in a library based environment. There remains considerable work in development and refinement of modeling standards to support the diverse design views of RASSP. This is especially true to information types being addressed under the CALS/PDES data umbrella. Work is ongoing in these areas, but increased coordination insure interoperability is key in addressing the creation and management of data views important to a RASSP design. Emphasis needs to be placed on unsupported model view under RASSP to address areas of performance modeling and systems requirements standard formats. RASSP long term efficiency in providing model views for design is related to design standards harmonization and compliance to a standard sets of protocols and data formats, such as those being advocated through CALS.

Development and infrastructure in hardware CAD design would immediately benefit from modeling standardization efforts. This would facilitate development of systems level and analog design automation tools and the supporting concurrent engineering processes. Model data standardization would facilitate new tool development and reduce risks for commercial

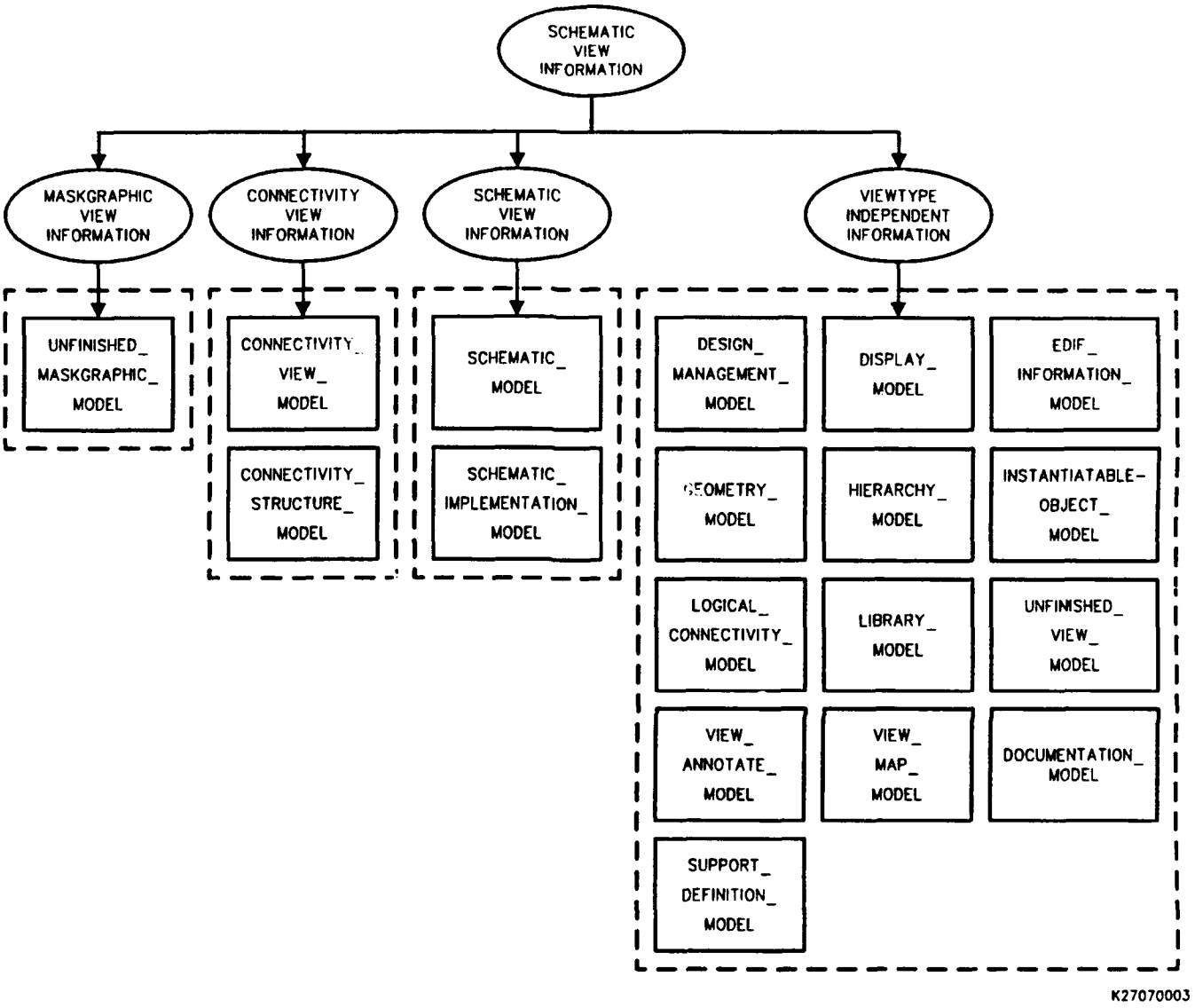


Figure 40
EDIF Information Model Views

tool development in these areas in much the same way that VHDL has served as a facilitating standard for digital analysis tool development.

Tools that address mixed analog and digital simulation are needed to support multiple levels of design and analysis of both pre-processing and data acquisition, and sampling sub-systems in a RASSP design. Analog design is still primarily at a low level of (SPICE) abstraction due to the lack of analog modeling tools and analysis capabilities at more abstract levels. Analog descriptions must be supported for RASSP system design in conjunction with digital design descriptions. Analog extensions to VHDL should be addressed along with mixed mode simulation tools that can support these standards.

There is a lack of robustness and integration in both data management and analysis tools to support concurrent engineering and statistical design analysis processes. This is especially true for component level model views that may be supplied by component vendors. The development of a new generation of concurrent engineering capability and statistical analysis tools able to leverage off of standardized modeling and data formats such as VHDL and PDES is key for RASSP design. This would include standardized functions and attributes to address design and component tolerances in different model views.

The development of supporting standards for RASSP related design views is a priority for library development. Standardized function packages to support VHDL and other modeling standards are needed to allow reusable library views of DSP components and modules. These include extensions to modeling formats to allow the inclusion of performance and parametric information (such as PDL extensions for VHDL) for RASSP analysis.

The emphasis on a library based design flow for RASSP design points to the need for improvements in the design automation process and management. The need for development of design automation based librarian concepts, and the development of efficient means of managing and distributing library data between vendors, users, and suppliers needs to be refined from current levels. This is especially true with regard to management of libraries in a concurrent engineering environment that will be able to address the diverse model views and design types that are foreseen for RASSP.

Vendor part models and vendor participation in library development are critical to a RASSP development effort. Vendors need to be able to provide model views and information to a RASSP library in an accessible, but secure, means that will protect any proprietary information. As a preliminary step to development of such libraries, RASSP contractors need to define and implement the support mechanisms for common and interoperable libraries and data bases for RASSP vendors/users. Component and subsystem vendors of RAASP relevant hardware need to be active contributors to such library development efforts. Useful model information is a limiting factor on quality of a RASSP design library. In most current design efforts, the limiting factors in analysis are part model availability and quality. This is expected to be an equally critical issue in the RASSP hardware CAD environment.

7. MANUFACTURING

7.1 Foundry Interface/CAD

Section Flow: The Foundry Interface/CAD discussion is divided into two parts as shown in Figure 41. The first part, titled "Concept", discusses the status of the TI MCM Merchant Foundry and describes long-term RASSP concepts for the design-to-foundries interface and the foundry CAD system implementation. The second part, titled "RASSP Program", details the approach for demonstrating the proposed concepts and lists associated issues.

CAD

- SUN computers in place, 2 copies of Finesse installed, GE software installed.
- Two CAD designers trained; designs in work.
- CAD-to-Manufacturing data transfer software complete; validation under way.

TEST

- Three levels of test under way.
  - TEG tester: In validation
  - Mfg defects tester: In integration
  - Performance tester: On line

MANUFACTURING PROCESS

- Facility in place and complete.
- Equipment installation 90% complete.
- 4 Phase process transfer in work.
  - Ph 1: Photolithography @ 100%
  - Ph 2: Metalization @ 90%
  - Ph 3: Lamination @ 80%
  - Ph 4: Laser image/drill @ 50%

PRODUCIBILITY

- Key capability developed at transfer.
  - Electrophoretic photoresist
  - Multilayer lamination
- Cost reduction ongoing.
  - Pre-molded cavity substrates
  - Integral package development
  - Planarized capacitors
  - Low cost dielectrics

Figure 41
Presentation Flow

MCM Merchant Foundry Status: Figures 42 and 43 show that TI's MCM Foundry concept, implemented jointly by TI and GE under DARPA's Technology Transfer Program, is in agreement with DARPA's RASSP objectives and has the potential for use in fabricating RASSP products.

The CAD-to-Manufacturing data transfer was implemented using GDSII Stream data which is output from every major CAD system. Using a specification jointly furnished by TI and GE, the Harris Corporation has implemented GDSII formatting enhancements which identify physical information for subsequent design rule checks (DRC). Also, this GDSII data is passed to manufacturing where it is translated as input to operations such as substrate milling and laser drill and patterning. The specification for the enhanced GDSII was given to other CAD vendors for consideration as they upgrade their CAD systems for closer ties to manufacturing. Electrical information is passed to subsequent electrical rule checks (ERC) by a Spice-formatted net list which is also ad hoc industry standard.

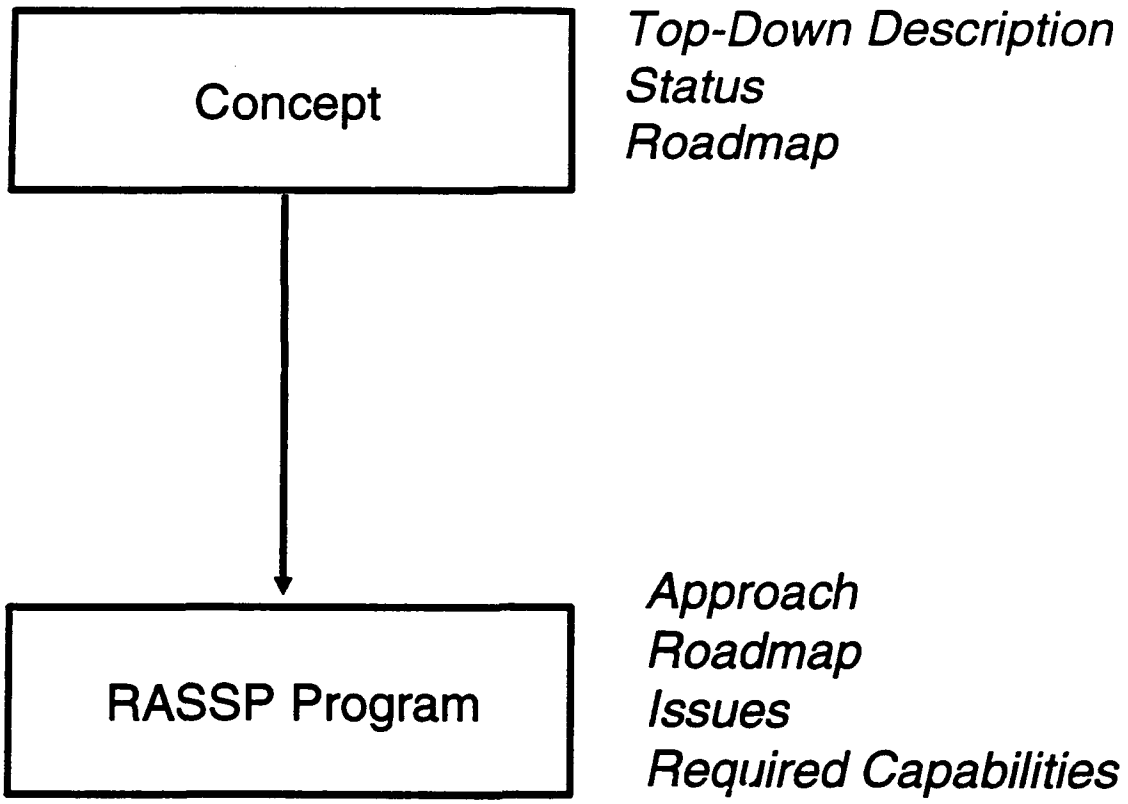


Figure 42
MCM Merchant Foundry Status

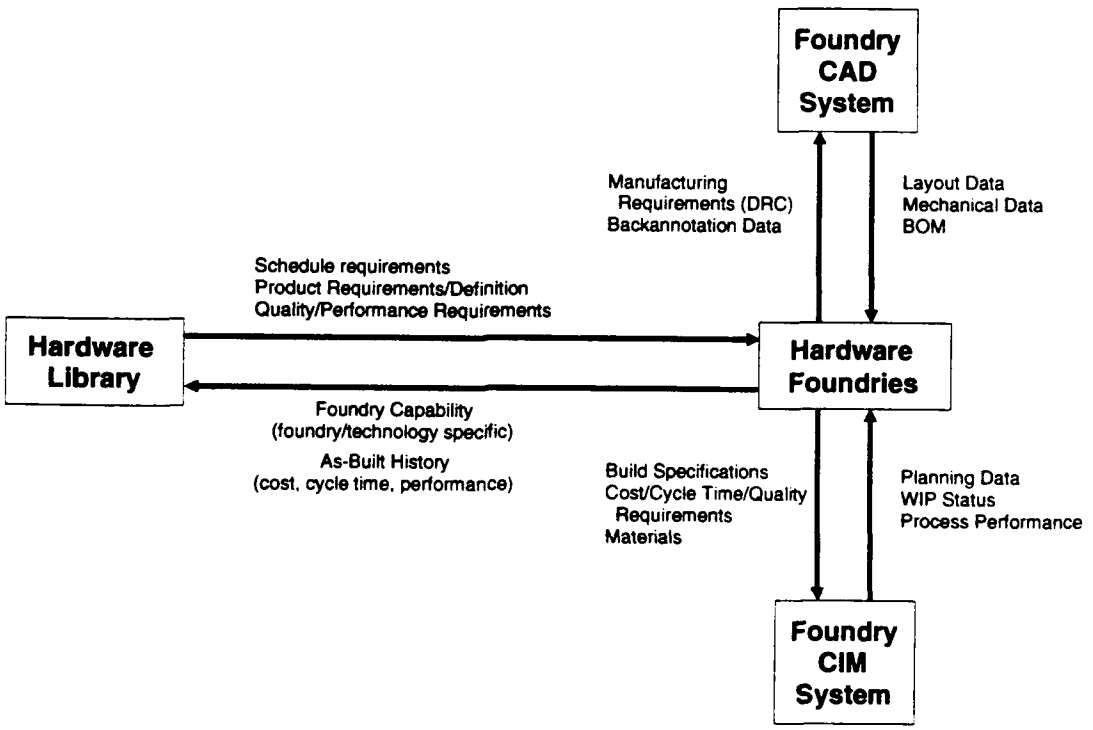


Figure 43
Hardware Foundries

Three levels of test were implemented in the TI Foundry. Supporting these levels are (1) a test element group (TEG) tester, (2) a manufacturing defects tester , and (3) a performance tester. Special attention was given to insuring that the test capabilities were implemented by purchasing vendor-supplied systems, such as the HP 82000 performance tester, or by configuring systems using industry standards such as the VME-Extended-Instruments standard (VXI). Boundary scan techniques, well known throughout the industry, also play a key part in the foundry's test strategy. Thus, RASSP concepts that support multiple levels of test and use available standards exist within TI's MCM foundry test operations.

In the MCM foundry, the manufacturing process is electronically tied to the design and layout process. This feature increases productivity by insuring that data consistency between design and layout is maintained, and it contributes to a decrease in cycle time by avoiding data entry at the foundry level. A significant example of this data transfer occurs between the CAD system, Harris Corporations's Finesse, and the laser drill and patterning system (LDPS) which is TI-fabricated. CAD data output in the GDSII format is translated into data compatible with software running at the LDPS which controls the laser during the complex drilling and patterning operations.

Several key producibility enhancements were outlined by TI and GE in the DARPA-sponsored Silicon Wafer Advanced Packaging Program (SWAP) Report. They include cost and cycle time reduction items such as pre-molded substrates and integral packages. These improvements are key to RASSP success because they will decrease the overall cycle time for RASSP users.

Opening up a Foundry: The "Open Foundry" is a concept initiated by DARPA, and it has been emphasized as the TI MCM Foundry has come on line. From the foundry CAD standpoint, an Open Foundry has three premises: (1) The foundry must be able to accept input from any CAD system, (2) CAD systems must have the capability to support multiple technologies, and (3) the foundry must support customer entry at multiple levels of the design process. These premises support the RASSP vision. Further insight into the Open Foundry concept is shown in Figure 44. This is an insert from an article by Tony Mazzullo, Vice President of Marketing of the Harris Corporation, that appeared in the Summer, 1992, issue of "Advanced Packaging" magazine, titled "CAD Software." It illustrates TI's commitment to the DARPA-initiated Open Foundry concept. Harris Corporation markets the Finesse CAD System used in the TI Foundry.

Hardware Foundries: High-level data requirements for interfaces to and from the foundry were shown in Figure 43. Figure 45 shows how these interfaces are implemented.

Schedule requirements, product definition, and quality/performance requirements flow from the RASSP user's hardware library to the foundry. Depending on the level of interface to the foundry during the design process, this information will be transferred via enhanced standard formats or combinations of formats as VHDL, EDIF, IGES, WAVES. Enhancements are needed in these formats to support test vector transfer, multiple MCM technologies, and business requirements such as schedule, cost, and yield information. An important characteristic of the design-to-manufacturing data flow is that, in addition to design data, data describing the RASSP user's business requirements is transferred to manufacturing.

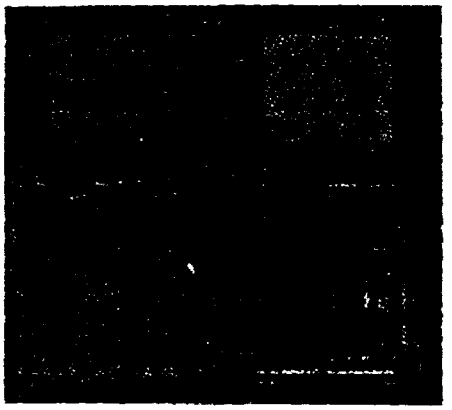
Foundry capability and as-built history flow from the foundry to the RASSP system. No standards exist for this critical flow which

quickly by redefining the process rules and performing place and route against the new rules. This may also be applied by taking an existing PC board implementation and converting the circuit to one of many alternate MCM technologies.

A breakdown of the process rules necessary for MCM design highlights many of the features required in a software package. Some of the process rules that drive the layout functions include:

- Database resolution
- Trace widths and clearances
- Feature sizes for vias, pads, traces, etc.
- Via stacking
- Process build order
- Chip assembly technology

Via stacking rules must support many different requirements including the most complex structures such as through-circuit (stacked), buried, blind, staircase, and staggered. An MCM/CAD system should also support complex rules where a limited number of adjacent vias may be stacked and/or vias may stack on alternating layers. The dimensions of the staircase, including minimum and maximum stagger distance, must be followed for place, route, edit, and design rule checking.



DSPIV Module This high performance MCM was designed by GE as part of the DARPA-sponsored "Open Systems for Foundry" concept. The CAD system used was able to integrate its design tools into existing foundry demands.

Other Considerations

Library data must accommodate a wide range of bare die and surface mount components. Many different assembly techniques requiring complex padstack with many of the same rules as vias must be supported. This includes via templates to describe pre-defined fan-out and fan-in patterns and pin-to-pad mapping for accurate correlation with netlist formats and substrate patterns.

Placement algorithms must be optimized for packing components tightly, based on assembly technology. Chip-to-chip spacing and placement optimization will vary, whether using wire bond, TAB, chips first, flip chip or other approaches.

The automatic router must be proven for MCM technology. The autorouter

OPENING UP A FOUNDRY

Texas Instruments (TI) and General Electric (GE) are teamed under a DARPA contract to transfer GE's "chips first" technology — High Density Interconnect (HDI) — to TI where a high-volume merchant multichip module (MCM) foundry is being established.

At past project reviews the DARPA review team has strongly emphasized its desire for the foundry to have the capability of interfacing with multiple CAD systems — thus, the term "open foundry." Reasons for establishing this criteria follow:

- It increases the foundry's customer base and subsequent volume, resulting in lower overall MCM unit costs.
- Customers can use a CAD system tailored to their design requirements — low-speed digital, high-performance, analog, etc.
- From a customer's perspective, it allows avoidance of the high cost of replacing an existing CAD system, thus encouraging the use of an MCM.
- It encourages the establishment and use of standards: design languages, framework, interface formats, user presentation, and communications.
- The use of "paperless" interfaces to the foundry is facilitated.
- It reduces the foundry's dependence on the continued existence of a given vendor and his support of the MCM technology.

The open foundry concept has two implications, one from the foundry perspective and one from the CAD system perspective. The foundry's automation systems must be capable of direct CAD system interfaces. Also, the foundry is forced to establish close relationships with vendors, consortia, universities, and standards committees.

CAD systems must support MCM requirements by including enhancements to routers, design rule checkers, concurrent engineering tools, schematic capture systems, part modeling concepts, and design languages. Most importantly, links with IC and ASIC systems must be established.

To meet this challenge, the TI/GE team has developed a strategy for developing an open foundry. The elements are:

1. Document and exercise current CAD system interface capabilities. The CAD system used within the foundry environment has various levels of capabilities for interfacing with other CAD systems. This allows the generation of foundry-specific manufacturing data.

2. Establish parts list/Gerber capability with vendors having a major portion of the market. Vendor's PC board CAD systems have this output capability and no further modification is necessary. Further supplements to this data are required at the foundry level.

3. Establish direct file-level interfaces to the foundry. This close coupling will reduce overall cycle time as well as reducing the chance of errors caused by miscommunication across the design/foundry interface. Also, no foundry automation enhancements are necessary.

4. Implement foundry interfaces via standard formats. This long term objective requires enhancements to existing standards supporting MCM technologies. It will make the interface across CAD systems consistent and encourage stability of software supporting this type of interface. —MARK ESKEW

Figure 44
DSPIV Module

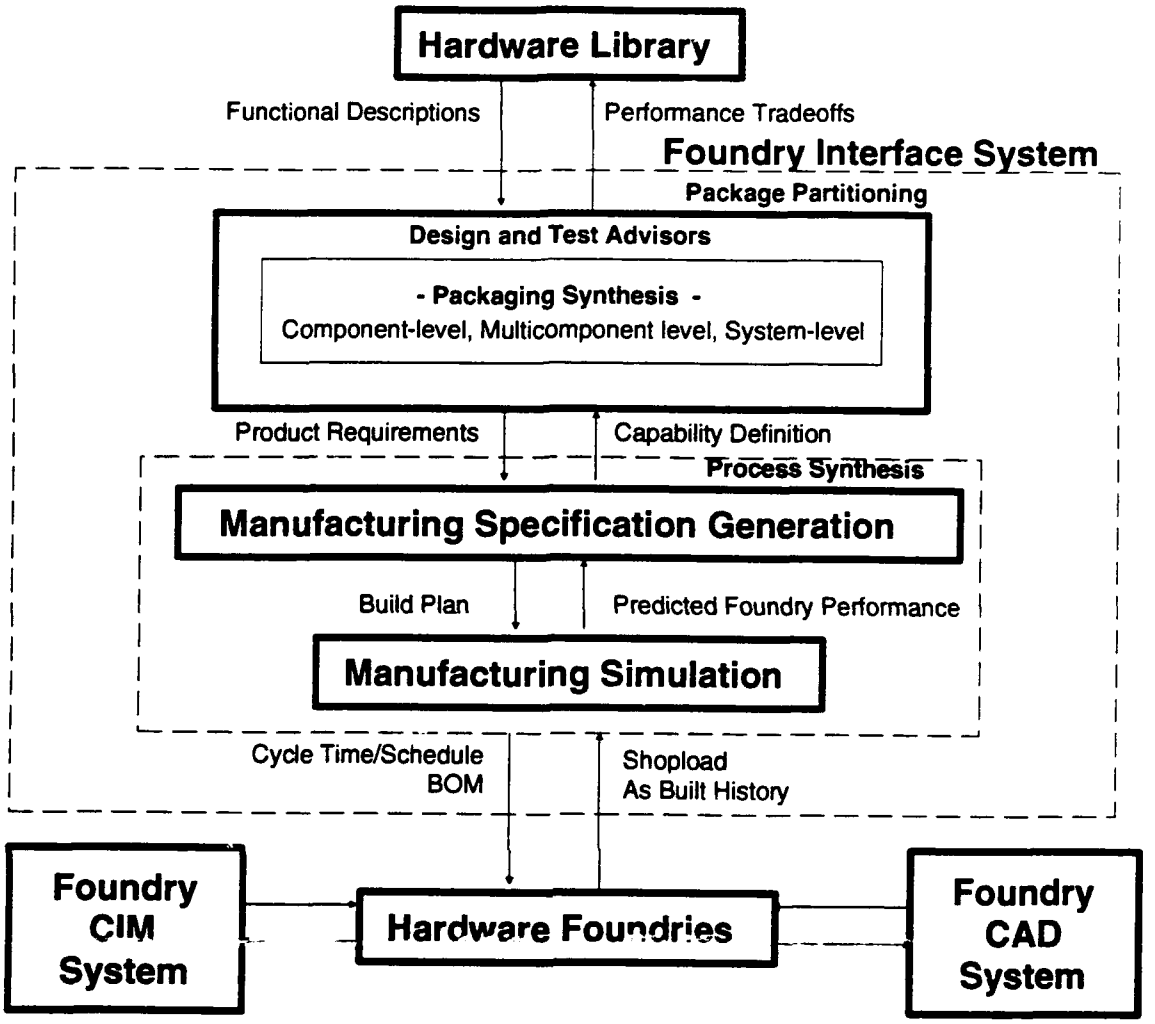


Figure 45
Foundry Interface System

identifies design changes necessary for process improvements. MCC's DARPA-sponsored "Infrastructure" effort will address these shortcomings, and TI will participate in this effort. The transfer of foundry data back into the RASSP design system is critical to the success of the concurrent engineering process. Testability, reliability, manufacturability, and other analysis tools need this feedback for continual improvement of their ability to forecast the performance of the "as-built" product.

Because this two way design-to-manufacturing interface is so critical to the RASSP vision, the lack of industry standards supporting all RASSP requirements must not stop the near-term implementation of this close-coupling. RASSP foundries must establish interfaces to RASSP design systems by individual agreements using ad hoc data standards such as GDSII or proprietary formats as established by CAD vendors.

Flowing from the foundry to the foundry CAD system is information specifying manufacturing design requirements. Manufacturing design requirements will determine the content CAD system's technology files which are used to insure correct by construction design in the layout

system. Also transferred is data describing manufacturing changes that will affect product descriptions residing in the CAD data base. Backannotation of this manufacturing data into the foundry CAD system is required to keep manufacturing data synchronized with CAD data.

Layout data driving the manufacturing process, mechanical data, and bill of material data are forwarded from the CAD system to manufacturing. In the TI MCM Foundry, this link is well-established, and CAD data is converted to drive the manufacturing process. However, in the past, this hand-off was manual, and some foundry operations remain in this mode.

The foundry CIM system is another key to successful RASSP productivity improvement. It must communicate planning data, WIP status, and process performance information to the foundry. In return the foundry transmits build specifications, cost/cycle time/quality requirements, and materials information to the CIM system. Mostly proprietary data exchange formats are currently used for this data exchange. In the future, data exchange standards are required. CIM system details are described in another section of this report.

Foundry Interface System: To establish an efficient interface into RASSP foundries, a "Foundry Interface System", depicted in Figure 45, is proposed. This interface system resides at the RASSP user's site and enables the user to select packaging technologies for implementation of his design, simulate manufacture, and transfer data to multiple foundries. The Foundry Interface System contains two major elements: Package Partitioning System and a Process Synthesis System.

The Package Partitioning System is composed of a suite of design advisors that aid the RASSP user in making packaging decisions at the component, multicomponent, and system levels. Microelectronics and Computer Technology Corporation (MCC) has DARPA and corporate-sponsored programs active in this area. MCC forwarded a tool status summary in the test advisor area (shown in Figure 47). MCC states that design and test advisors as well as synthesis tools will exist in the 1993-1994 period that have the capabilities needed for a demonstration of the Package Partitioning System. MCC supplied a chart (Figure 48) illustrating this concept, and Peter Sandborn of MCC provided the following write up on this topic.

> "Figure 48 shows a picture of a packaging (hardware) synthesis system which could be implemented for RASSP. The system is focused on three primary activities: Component (chip) level synthesis/specification, module/board (packaging) level synthesis/specification, an multiple board (system) level synthesis/specification. The component-level activities must support synthesis of systems from a mixture of off-the-shelf an ASIC components. The off-the-shelf components are assumed to be described in the Hardware Library. The number and type of discrete components needed to realize a system is a function of the architecture and packaging and must be concurrently addressed at the component and multicomponent levels. Again, a library of off-the-shelf discrete components should be available in the Hardware Library. View specific design advisors are treated as shown in Figure 49 (an advisor consisting of analysis and evaluation pieces). The synthesis rectangles in Figure 48 are the design advisor manager pieces. A common procedural interface between the specification/synthesis tools and advisors must be developed so that advisors can be reused by various parts of the hardware synthesis system an design cycle. The reuse dictates that advisors be "hierarchical" whenever possible so that their

MCC MICROELECTRONICS AND COMPUTER TECHNOLOGY CORPORATION

Summary of Tool Status

Known tradeoff analysis tools:

Tool/Vendor	Key Capabilities	Link to Other Tools	Language Platform
SUSPENS/Stanford University	• Simple chip-level physical tradeoff analysis • Obsoleted by tools below	None	?
MICON (Fidelity)/CMU (Omniview, Inc.)	• Physical and behavioral synthesis of multichip systems from off the shelf component library • MICON supported single board design only • Omniview commercializing MICON	Synopsis - ASIC synthesis	? MICON C++ Fidelity
System Design Planning Tool/NTT	• Extends MICON to multiboard designs • Not publicly available	?	?
PEPPER/IBM	• Constraint driven (delay) pre-router focusing on functional partitioning into chips and chip placement • Pre or post-netlist operation • Not publicly available	?	C
AUDIT/Cornell University	• Chip and module-level physical tradeoff analysis at a hypothetical level • Global optimization using a simulated annealing approach • SRC funded	Gnuplot	C/X windows
YODA/CMU	• Architectural/algorithmic tradeoffs for digital signal processors • Well developed design advisor/knowledge-base philosophy	None	LISP
SPEC (MSDA)/MCC	• Detailed module-level physical tradeoff analysis for real modules • Commercially funded consortia project • Mentor, Cadence, Intergraph developing commercialization plans	SPICE 2D electrical simulation (FEM)	C++/X windows

Figure 46
MCC Summary of Tool Status - Known Tradeoff Analysis Tools

view of the design state is as independent of the level of the design hierarchy as possible.

The key elements of the system are the feedback between the different synthesis levels. Without this mechanism the specification and synthesis activities are little better than a traditional serial design process. Chip design (or selection) must be accomplished concurrently with packaging technology selection if optimum solutions are to be reached. An additional key element is the link between the packaging synthesis environment and the set of tool and advisors shown on the right hand side of Figure 48. Packaging synthesis can not be accomplished without close linking of ASIC synthesis tools (existing ones) and detailed thermal and electrical simulators. We have also shown the view specific advisors mentioned above and shown in Figure 49 as a library into which new advisors can be added in the future. The domain specific advisors is a knowledge-base of known design practices for subsystem design."

It is proposed that the Package Partitioning System be implemented and coupled with DARPA-sponsored MMST tools to form an overall Foundry Interface System (Figure 45). MMST tools will give RASSP users the capability of predicting foundry cost, cycle time, and yield before

MCC MICROELECTRONICS AND COMPUTER TECHNOLOGY CORPORATION

Summary of Tool Status

Test Advisor Tools Versus RASSP Requirements

	TDES	DEFT	Shirley's	Design Expert	Intelligen	TIGER
System-level Testability Analysis	No	No	No	No	No	No
Cost/Benefit Trade-offs	Yes	Ye	No	No	No	Yes
Design Partitioning	Primitive	Primitive	Primitive	No	No	Yes
Alternative Test Methodologies	Yes	No	No	No	No	Yes
Prioritization of Solutions	Yes	Yes	No	No	No	Yes
Constraints Satisfaction	No	No	No	No	No	Yes
Cost Estimates	Yes	No	No	No	No	Yes
Flexible Technology support	No	No	No	No	No	No
Embedded macros	Yes	No	No	No	No	Yes
RTL Test Sensitization and Propogation	Yes	Yes	Yes	No	No	Yes

Figure 47
MCC Summary of Tool Status - Test Advisor Tools

commitment to foundry builds. Also, MMST tools will use the product description to generate a manufacturing specification that contains the initial foundry routing information. MMST personnel have doubled this manufacturing specification generation and simulation effort "Process Synthesis".

Jack Mahaffey, who holds TI responsibility for Computer Integrated Manufacturing (CIM) on the DARPA-sponsored MMST Program, supplied the following description of the two MMST components that are linked with the Package Partitioning System to form the Foundry Interface System as shown in Figure 45.

> "Designing for producibility requires an ever increasing need to tighten the coupling between the design and manufacturing processes. The proposed Foundry Interface System supports this need. The foundry interface consists of packaging partitioning tools and design and test advisors coupled with DARPA-sponsored MMST tools enhanced to support manufacturing specification generation and foundry simulation.
>
> The package partitioning tools will provide the product requirements at the component, multicomponent, and system levels. Multicomponent product requirements will then be fed to the MCM foundry.
>
> A manufacturing specification for the MCM products will be generated by tailoring the standard MCM product routing based on

the unique processing requirements of the MCM product. At each step in the manufacturing process, the step specification is tested against process models which reflect the capabilities of the manufacturing equipment. The specifications are adjusted to optimize yield at each step based on process simulations.

The manufacturing specification will be used by a stand alone factory simulation tool to predict cost, cycle time, and yield through the foundry. The foundry simulator is based on a foundry model, or representation of the foundry resources (people and equipment), historical yield and cycle time distributions, and equipment, labor and consumable material cost information. Additionally, the foundry shopload must be specified in order to predict product cycle times. The foundry model is populated and updated using a series of foundry model editors provided with the tool.

The simulator captures an "executable instance" of the populated foundry model and runs a discrete event simulation of the production build sequence for the product, based on the manufacturing specification. The simulator provides a "what-if" capability, allowing users to make specification trade offs before committing the design to manufacturing."

Data interface standards for communication of information between the design function, package partitioning, and the manufacturing function, process synthesis, do not exist. It is proposed that MCC's DARPA-sponsored "Infrastructure" effort be the catalyst for this effort, and that, if necessary, interim ad hoc or proprietary standards be used for the RASSP feasibility demonstration.

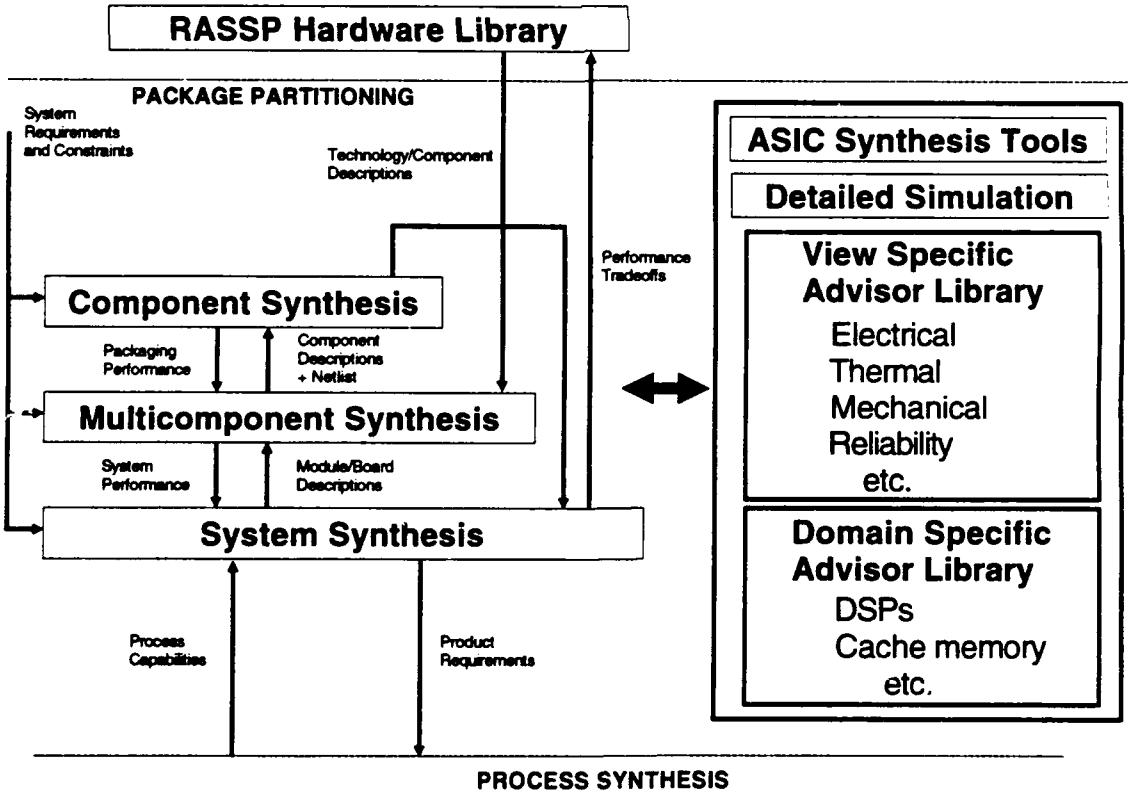


Figure 48
MCC Packaging Synthesis System

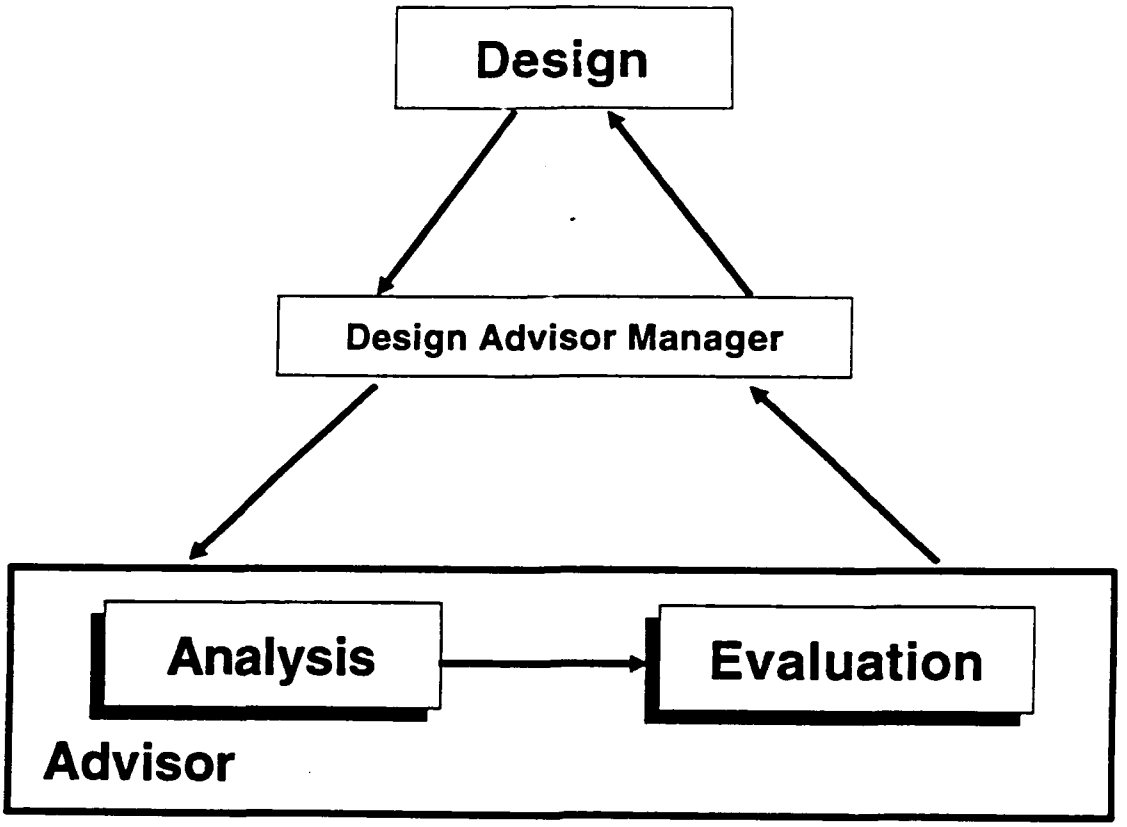


Figure 49
Design Advisor Components

CAD Activities: The CAD sources listed in Table 17 were contacted and plans related to the RASSP vision were discussed. The following common themes were revealed.

1. Vendors are working toward tying together CAD tools across packaging levels. For instance, Cadence, who recently absorbed Valid Corporation, is tying together IC and MCM design tools originating in the respective organizations.

2. Real-time analysis tools (testability, reliability, manufacturability, etc.) are scheduled for release by the vendors in the 1993-1994 time period. All are aggressively pursuing this feature which is key to supporting the concurrent engineering process. Mentor Graphics (MGC) and TI recently submitted an ASEM proposal to accelerate this effort within MGC's MCM Station CAD system used by many potential TI MCM foundry customers.

3. Vendors are adding MCM CAD systems to their marketable items. Thermal analysis, transmission line analysis, and layout improvements required by specific MCM technologies are examples of enhancements vendors are including with their MCM systems.

4. Implementation of CAD-to-Manufacturing interfaces are underway. For example, Harris Corporation, has implemented a "smart GDSII" CAD output, using a TI/GE supplied specification, that is capable of driving design rule checkers and the TI foundry laser drill and patterning system.

Table 17 CAD Activities

CAD Sources	Activity	When	RASSP
Mentor Graphics	MCM technology specific enhancements.	3Q93	3Q93
	Integrate IC/BOARD/MCM Station.	1993-4	1Q94
	Couple place and route with real time analysis.	1994-5	1Q94
	Extend DRC/ERC (Checkmate) for MCM.	2Q93	3Q93
Harris	Produce "smart" GDSII for DRC/ERC and manufacturing interface.	Avail	3Q93
	Implement MCM technology specific enhancements.	Avail	3Q93
	Develop generic database to interface with any CAD tool.	1994	2Q94
	Couple place and route with real time analysis.	1995	4Q93
Cadence	Integrate IC (Cadence) and PWB (Valid) CAD systems.	1993	2Q94
	Implement MCM technology specific enhancements.	3Q93	3Q94
Dasix	Continue "Advanced Packaging Program."	Avail	N/A
	Implement thermal/electrical driven CAD system.	1994-5	1Q94
MCC	Execute ASEM infrastructure contract.	1Q94	1Q94
	Implement Design Advisor.	1Q94	4Q93
	Execute "Known Good Die" effort.		
	Continued involvement: Electronic Data Book, DFT, Enterprise Integration Network.		
Mayo Clinic	Fan out DARPA-sponsored HF Analysis tools.	2Q93	4Q93
U of Maryland	Extend CALCE tool set for MCMs.	1Q94	4Q93
U of Arizona	Commercialize HF Analysis tools.	Avail	4Q93
U of Cincinnati	Synthesis enhancements for MCM	4Q93	3Q93

Research organizations and universities are contributing to the RASSP vision by developing and evaluating high-performance analysis tools, concurrent engineering enhancements for MCM, and synthesis enhancements for MCM. TI has contributed to MCC's ASEM Infrastructure proposal, is teamed with the University of Cincinnati for development of MCM synthesis tools, and is heavily involved with the University of Maryland's Calce effort and the extension of their tools to support MCM.

The last point regarding Table 17 is that all availability dates ("When") are in the range needed for the RASSP foundry CAD demonstration.

Foundry CAD System: Because RASSP foundries must support users who want to interface at multiple entry points in the design cycle (Open Foundry Concept), the foundry CAD system, shown in Figure 50, has functional and interface requirements corresponding to RASSP CAD systems discussed earlier in this report. This information will not be repeated, and the current discussion will focus on the characteristics of a foundry CAD system which set it apart from the RASSP user's CAD system. However, the foundry CAD system and the RASSP user's CAD system may be the same.

The foundry CAD system must be library driven, and thus, technology independent. The user may choose to implement a RASSP design in one technology and later redirect the design to another technology and foundry with no additional intervention. This gives the user flexibility from a business and a technical standpoint. Thus, foundry CAD libraries must contain design rules supporting multiple technologies and multiple foundries.

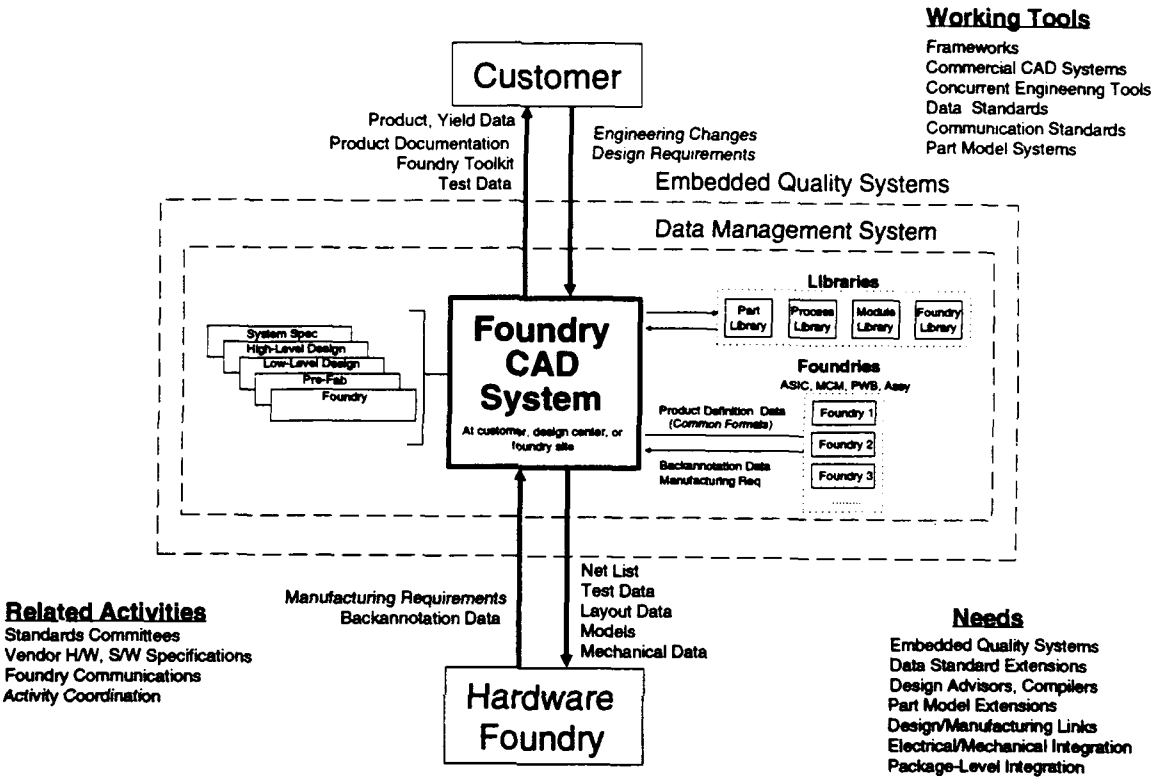


Figure 50
Foundry CAD System

The foundry CAD system must have the capability of communicating with multiple foundries. Transmission of design information to other foundries via standard formats is a requirement; and transmission of data from other foundries must be available to keep data bases synchronized and concurrent processes updated. That is, the capability of keeping the "as designed" and "as built" data bases synchronized is a requirement. Data standards for foundry-to-foundry interfaces must be examined for completeness and exercised for company-to-company data exchange. Standardization of information describing part, process, module, and foundry processes are essential to the RASSP thrust.

Required Capabilities for Foundry CAD: Table 18 summarizes the requirements depicted in Figure 50. The table gives required capabilities, their characteristics, associated issues, and risks.

The "Characteristics" column of the "Circuit Layout" row indicates that capabilities such as electrical and design rule checks, high performance circuit analysis, analog and digital simulation, and concurrent engineering tools must exist in a commercially available framework that is capable of accessing multiple packaging technologies. The "Issues" and "Risks" columns imply that vendors will not respond to RASSP

Table 18 Required Capabilities for Foundry CAD

Capability	Characteristics	Issues	Risks
Circuit Layout	1. Online ERC, DRC 2. Multiple pkg support 3. High perf cape 4. Exist in framework 5. Analog, digital support 6. Commercially avail 7. Online "-ilities"	1. Not currently avail 2. CAD sys pkg specific 3. Cape not lab validated 4. Vendor specific, no stnd 5. Analog req unsupported 6. Market driven 7. Not currently avail	1. S/W performance 2. Rulebase gen, support 3. Validation expense 4. Untried in comm env 5. Algorithm devel 6. Lack of business 7. Rulebase gen, support
Dsgn/Layout data I/F	1. Support mult tech 2. Support dsgn hier 3. Bidirectional 4. Stnd formats	1. Not currently avail 2. Avail within tech 3. Very limited cape 4. Need tech extensions	1. CAD tech separate 2. Cape exist 3. CAD/CAM vendor team 4. Acceptance
Layout/Mfg data I/F	1. Standard I/F formats 2. Bidirectional 3. Foundry support	1. Only defacto exist 2. Very limited avail 3. Foundry specific now	1. Agreement, accept 2. CAD/CAM teaming 3. Low risk, in work
Dsgn/Test data I/F	1. Test data mgmt system 2. Bidirectional 3. Tied to CAE	1. Only point solutions 2. Very limited avail 3. One way interface	1. Vendor support 2. CAE/CAD/CAT team 3. Concurr tool avail
Product defn database	1. Object oriented 2. H/W, S/W indep 3. Content specified	1. Immature tech 2. Difficult to implement 3. Comm/DoD agree	1. Maturity 2. Performance 3. Completeness
Backannotation to dsgn	1. Elec, mech, test 2. CAD system compatible 3. Concurrent engr integ	1. Disciplines not integ 2. Vendor breadth lacking	

requirements until the market demands the associated improvements. DARPA funding is key to encouraging earlier development of RASSP capabilities.

Major shortcomings exist in the availability of adequate standard data formats capable of transferring information from design-to-layout, layout-to-manufacture, and from design-to-test. Formats are either proprietary or ad hoc standards. Examples are vendor-offered formats, Spice, GDSII, and DXF. DARPA-sponsored ASEM Infrastructure Projects will address defining specifications for extensions to standard formats such as VHDL, EDIF, PDIF, WAVES, etc. Associated committees will then adopt the recommendations. Not only is the implementation of these standards critical, but the infrastructure necessary for their continued support is also required.

However, RASSP foundries must not wait on the definition, approval, and implementation of these standards and their support within commercially available systems. RASSP foundries must define and sponsor interim standards with enhancements meeting their requirements if productivity goals are to be met in the near-term. The TI MCM foundry has led the way by specifying GDSII formatting conventions, implemented by Harris Corporation, capable of spanning the gap between design and manufacture.

Another item critical to RASSP success is the product definition data base. Whether one or many coordinated data bases, this concept is key to data communication between all parties involved in a RASSP build. Data base concepts presented elsewhere in this report must include foundry requirements in their specifications. This insures that RASSP user requirements are met and that concurrent engineering concepts tying design and manufacture are implementable.

Finally, the concept of backannotation from manufacturing to design is required. This enables design improvements through concurrent

engineering tools that result in quality, cost, and cycle time improvements. Another important reason for this interface is to keep manufacturing data bases synchronized with design data bases. Occasionally, changes to design are required at the manufacturing level because of process changes, equipment replacement, or to compensate for a lacking CAD capability. Changes must then be reflected in the RASSP design data base which is the "master".

Foundry CAD Demonstration for RASSP Program: A demonstration of the Foundry Interface System is proposed. An overview of this proposal is shown in Figure 51. Team members will assemble a hardware library based on an existing RASSP design. The library will consist of software descriptions of ICs, MCMs, and subsystems. The library will include HPC, DP, interconnect, and sensor interface modules. The group will describe the library using of standard formats such as VHDL, IGES, EDIF, when possible or will default to ad hoc alternatives such as vendor proprietary formats, if not possible.

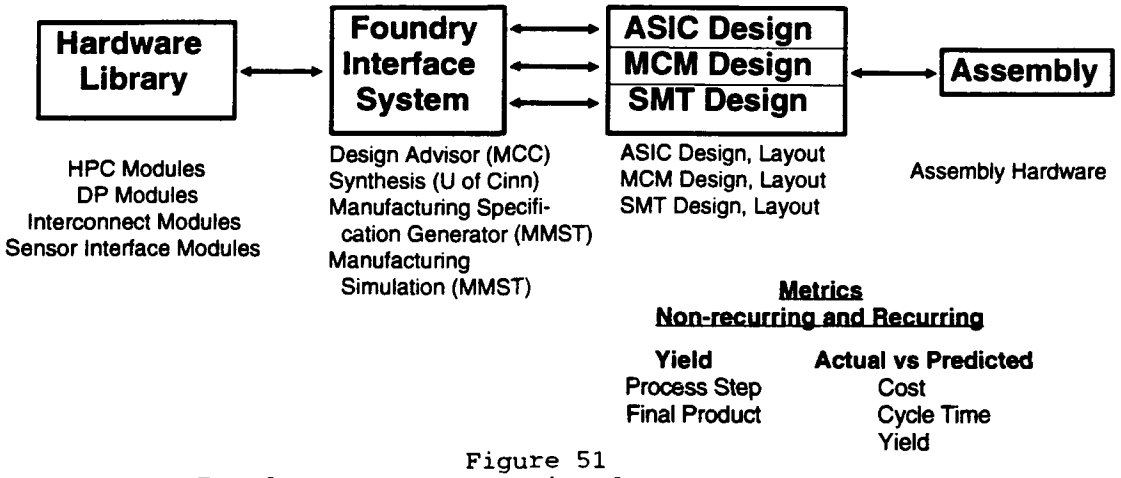


Figure 51
Foundry CAD Demonstration for RASSP Program

Working with MCC and associates who are participating in the participant-funded Design Advisor and DARPA-sponsored Infrastructure Projects, a system of design and test advisors and synthesis tools will be assembled forming a Package Partitioning Function. It is likely that a full complement of design advisor and synthesis tools to support the demo will not be feasible within the time and financial limits of the demonstration. Project personnel will implement an adequate number of the software tools to demonstrate feasibility of the approach, and they will supplement automated procedures with documented, manually implemented procedures.

TI MMST personnel will work with MCC in interfacing the Package Partitioning Function to MMST's Manufacturing Specification Generation and Manufacturing Simulation tools. These tools are operational in TI's IC manufacturing areas, and it is anticipated that the effort to move the tools into a RASSP environment will be nominal. The major effort is to determine and mutually agree on the data interface specification between the design part of the Foundry Interface System, package partitioning, and the manufacturing part, process synthesis. This effort will require significant participation levels by both TI and MCC.

After the Foundry Interface System is assembled, as stated before, an existing RASSP design, potentially represented by VHDL and block diagram data, will be processed by the Package Partitioning System into multiple RASSP components (eg., ASIC, MCM and SMT). Following package

partitioning, the Process Synthesis System will predict cost, cycle time, and yield data for the associated foundries. It will also build initial foundry routings for the RASSP components. Foundries will then process this data and build the components. At least one foundry external to TI will electronically receive product data and will process a RASSP component. A project demonstration goal is to exhibit TI capability for building multiple packaging levels of the RASSP design in a common foundry.

Productivity metrics will quantify the effectiveness of the system. For both non-recurring and recurring functions, project personnel will capture data comparing process step and final product yield. Statistical analysis techniques will determine conclusions from the two measurements. Also, for both non-recurring and recurring functions, actual versus predicted data will quantify the effectiveness of the design advisors, concurrent engineering tools, and manufacturing simulation steps. The demonstration will show that this data, backannotated into design tools, will yield improvements in manufacturing.

Foundry CAD demonstration for RASSP Program (Plan/Install/ Demonstration): Table 19 shows a plan supporting the proposed RASSP demonstration. In the two to three month planning phase, the demonstration approach is fully defined. Detailed plans are completed, enhancements to MCC and MMST tools are identified, personnel and hardware resources are defined, project team members are identified, and a concurrent engineering approach is documented.

Table 19 Foundry CAD Demonstration for RASSP Program

Plan (2 - 3 months)	**Install** (9 - 12 months)	**Demonstration** (6 - 9 months)
• Complete Detail Plan	• Put HW, SW System in place	• Produce designs for system components
• Identify required enhancements: MMST, design adv, synthesis	• Produce data flow requirements	• Feed back designs to "customer" for analysis
• Identify required HW, SW pers resources	• Install, learn design advisor, synthesis tools	• Use concurrent engineering tools during design phase
• Identify project teaming	• Implement stand-alone ASIC, MCM, and SMT CAD systems	• Build product
• Identify concurrent engineering approach	• Specify integration approach	• Evaluate as-built against as-designed
	• Integrate tools per specification	

In the nine to twelve month installation phase, data and tools are put in place. Specifically, hardware and software systems are installed and personnel trained in their use. Existing RASSP modules are collected and stored in the RASSP library system for the demonstration phase. Data flow requirements are documented and used to specify an approach to system integration. Finally, personnel will integrate tools and form a system that supports the demonstration.

In the six to nine month demonstration phase, foundry builds occur and metrics are collected. One possible scenario is building ASICs externally, importing data and building MCMs and surface mount boards in TI foundries using a common CAD system.

RASSP CAD Issues: It is hard to prioritize issues related to this approach - all are key. However, leading the list, shown in Table 20,

Table 20 RASSP CAD Issues

ISSUE	APPROACH	MUST HAVE	BEING WORKED	WHEN	RECOMMENDATION
CAD-to-Manufacturing links #1	Harris supports "smart" GDSII, equipment vendors working with CAD vendors. IGES, PDES, WAVES, VHDL support in work.	Yes	Yes	1993-1994	Continued use custom interfaces until standards are supported.
Standard data exchange formats #2	ASEM Infrastructure projects addressing. Committees will approve.	Yes	Yes	1994-1995	Continue to use custom interfaces until standards are approved.
Design advisors #3	ASEM funding in place. MCC has initial version. Vendors are participating and will productize.	No	Yes	1994	Emulate design advisor and design packager and estimate productivity gains. Adopt ASEM software when available and measure.
Multipackage level CAD systems	Mentor, Cadence, Dasix integrating IC, MCM, and PWB CAD systems.	Yes	Yes	1994-1995	Use interfaces between packaging levels until S/W upgrades available.
IC, ASIC, MCM part models	Standards in work by committees.	Yes	Yes	1995	Continue to use currently defined models plus supplemental data.
RASSP Infrastructure	ASEM funding in place to address issues (consortiums, brokers,...)	No	Yes	1994-1995	Participate in consortiums: MCC, vendors, universities.
Real-time analysis (electrical/mechanical, high-performance)	Mentor Graphics, Cadence, Dasix, Harris pursuing in next generation CAD systems	Yes	Yes	1994-1995	Use standalone analysis in interim Implement new CAD releases when available.
Global CAD databases	University development.. Vendor development in area as part of system integration efforts.	No	Yes	1995 +	Continue using and coordinating multiple databases. Use frameworks to aid in coordination.

are definitions of data interchange formats for CAD-to-manufacturing and for foundry-to-foundry. Next is the implementation of a suite of design advisors supporting the packaging partitioning process. MCC and its participants will take the lead in defining and generating these functions. If some of the automated design advisor functions are unavailable, they will be performed using procedures generated by the project. In this context, the "Must Have" column is marked "No" for the total suite of automated design advisor functions.

Summarizing, tools will exist during the execution of this project for implementing a demonstration of the proposed Foundry Interface System. This system is a necessary component for realizing RASSP productivity goals. This exciting concept will provide links from design to manufacture that are currently nonexistent.

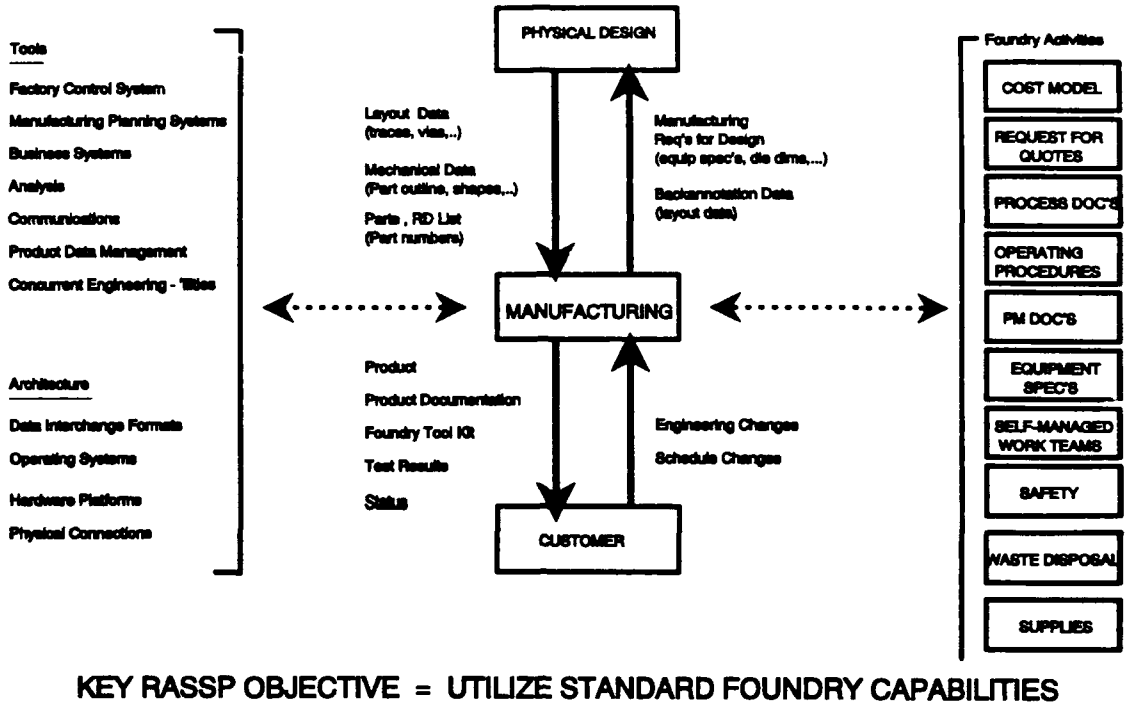
7.2 Manufacturing Overview

Figure 52 depicts some of the common elements associated with a RASSP program manufacturing environment. RASSP programs cannot be expected to be completely vertically integrated with all manufacturing and engineering capabilities in house. By necessity many of these programs must depend on a network of capable foundries to provide the manufacturing technology required to produce RASSP subsystems.

Each foundry must establish itself as a fully capable technology center with a full suite of manufacturing capabilities to support the RASSP program needs. The major elements of this capability include a well documented baseline manufacturing process, computer integrated manufacturing system, and a system of customer and design system interfaces.

Let us look at some of the elements of establishing the baseline manufacturing process.

The foundry physically is represented by the facility in which it is located. This facility contains the administrative and manufacturing areas in which the subsystems are constructed. This facility grew out of a business plan which comprehended the technology requirements of the



KEY RASSP OBJECTIVE = UTILIZE STANDARD FOUNDRY CAPABILITIES

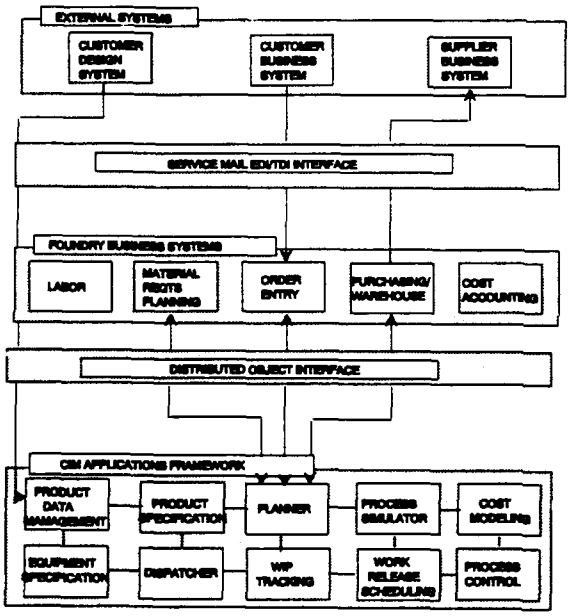
Figure 52
Common Elements Associated with RASSP Program Manufacturing Environment

products to be manufactured within the four walls of the foundry and a physical plan for implementing the technology. This business plan identifies market opportunities for the manufactured product. These opportunities are translated in detailed capacity and marketing plans. With these elements in place, the equipment and staffing plans can be developed. In parallel with the physical plant planning detailed analysis of cost, cycle time, and quality capabilities must be determined. These determinations allow the foundry to establish pricing models from which its marketing group will establish sales objectives.

A major effort in establishing the baseline process capability is equipment selection. It is extremely important that each piece of equipment be acquired to a documented set of requirements. Equally important is the documentation of the acceptance criteria for each of these requirements. The requirements must not only comprehend processing capability, but also conform to foundry environmental considerations, CIM architectural requirements, uptime goals, repairability objectives, and throughput objectives. Each piece of equipment should be brought online after successfully completing an acceptance test and further being certified to process capability objectives. These capabilities should be stated in universally understood variables such as Cp and Cpk. This effort will lay a foundation for continuous improvement as part of a foundry Total Quality Management Plan. Benchmarking of process capability is a major component of this plan and an excellent vehicle for ensuring the long term competitiveness of the foundry capability. Managing process capability is also a cornerstone of a Qualified Manufacturing Line (QML) approach to a quality reliability approach which we believe best satisfies Mil-Std requirements in a RASSP facility.

In addition to the physical equipment, a foundry contains significant amounts of documentation. This documentation is necessary to describe internal process procedures, operating instructions, preventative maintenance instructions, safety rules, environmental requirements, etc. Existence and utilization of this documentation ensures the consistency of operation associated with smooth execution within a capable foundry.

The foundry must be equipped with CIM tools. These tools must be deployed within an interoperable architecture to facilitate intra- and inter-communications with in house elements and customers. These tools span engineering, business, and manufacturing systems. They facilitate communication between the foundry and the customer, other foundries, and to the detail design systems. Information must flow freely both into and out of the foundry. Figure 53 depicts a sampling of these tools.



Required Capabilities

- Scaleable, distributed workstation based architecture
- Supports low to high volume. Planning with user selected goals: *cycle time, utilization, ...*
- Provides computer-aided decision support for planning, dispatching, material planning to meet delivery dates.
- Product specification and simulation capability *supports "what-if" analysis of various* operational decisions.
- Seamless ties to customer/supplier and business systems.

Figure 53
Tool Samples

Each of the foundries in the network must possess a capability to interface with both customers and the detail design systems. These design systems may reside within the foundry or external to the foundry. The interface between the foundry and these other elements will contain a bi-directional flow of information. The customer provides product information to the foundry such as schedule and engineering change information. The foundry, in return, provides the customer with tools to facilitate interface with the foundry and actual product history in return. The detail design systems provide the foundry with detailed product description data. In return, the foundry provides manufacturing requirements to the design system as well and back annotation data for individual designs.

A network of foundries is required to support the objectives of the RASSP program (Figure 54). Some elements of the manufacturing structure are mature and sufficient capacity already exists to meet RASSP's objectives; for other technology elements they are just beginning to emerge. Capacity is limited and significant barriers must be overcome to ensure the domestic development of these technologies capacities. MCM is an important example of this immature market. Technology capabilities are still being determined and explored. As a result, the supporting domestic infrastructure is immature and nearly non-existent. The business investment requirements are significant. The market window for these key emerging technologies is narrow. If the key capabilities are not put in place during the narrow window of opportunity, the domestic capability may never develop.

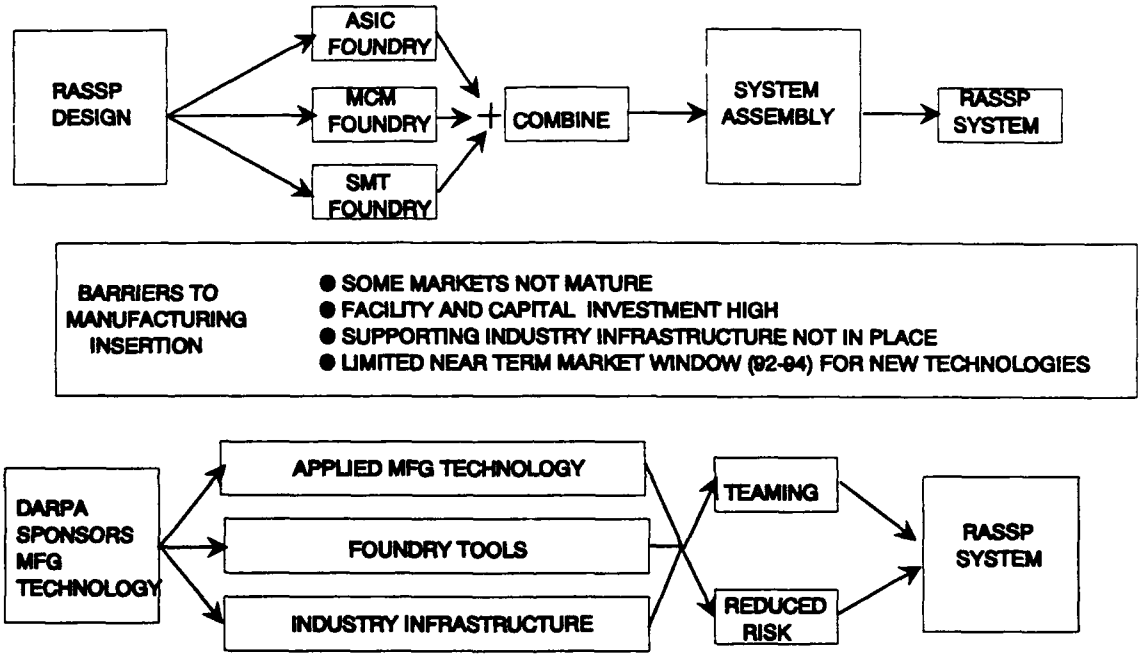


Figure 54
Network of Foundries

These newborn key technologies look to sponsoring agencies such as DARPA to act as a catalyst to bring together the cooperation necessary among the participants to create the critical mass to generate a base from which the foundries may develop. This sponsorship provides the jump start inertia to permit application of developed manufacturing technology capability through fan-out of technologies and cross licensing. Sponsorship of tool vendors permits the development of common tools which can be applied across the technology. Finally, the sponsor can act as the catalyst to bring together competitors to form industry associations to work solutions to common problems and barriers. An excellent example of this sponsorship is the DARPA MCM foundry program and the MCM technology association currently in the formative stage. Creating this environment will lead to teaming arrangements and reduced risk (cost, cycle time, and quality) to the customer.

With the functional foundry structure in place, the RASSP program may go to their yellow pages of contract foundries, Table 21, to select strategic partners for manufacture of components for their systems.

Table 21 North American Independent PCB Assemblers

COMPANY NAME	NUMBER OF EMPLOYEES	SIZE OF FACILITY	YEARS IN BUSINESS	IN-HOUSE MANUFACTURING CAPABILITY				
				PCB DESIGN	BOARD FAB	SMT ASSEMBLY	THROUGH-HOLE ASSEMBLY	FUNCTIONAL IN-CIRCUIT TEST
ADCO CIRCUITS INC.	70	22	11	•		•	•	•
ADVANCED ELECTRONICS INC.	300	65	16			•	•	•
ALTEK CO.	50	26	20	•		•	•	•
ALTRON INC.	185	65	18			•	•	•
AVEX ELECTRONICS INC.	1635	350	28	•		•	•	•
CORTELCO USA INC.	1100	351	30	•		•	•	•
DATA SIGNAL INC.	38	21	21	•		•	•	•
DIAGNOSTIC INSTRUMENT INC.	70	15	20			•	•	•
DOVER ELECTRONICS MANUFACTURING	425	125	23	•		•	•	•
ELECTRONIC SYSTEMS INC.	150	35	12			•	•	•
EMD ASSOCIATES INC.	600	105	18	•		•	•	•
EOG INC.	150	55	31	•	•	•	•	•
EVEREADY INDUSTRY CORP.	120	30	5	•		•	•	•
FLEXTRONICS INTERNATIONAL	600	44	21	•		•	•	•
FORCE INC.	52	24	14	•		•	•	•
4TH GENERATION SYSTEMS INC.	50	24	22	•		•	•	•
GENERAL TECHNOLOGY CORP.	160	35	5			•	•	•
GROUP TECHNOLOGY CORP.	850	308	27	•	•	•	•	•
HIBBING ELECTRONICS CORP.	400	90	18	•	•	•	•	•
VERN KIEBLER ELECTRONICS INC.	101	47	21	•		•	•	•
MANU-TRONICS INC.	250	74	22	•		•	•	•
MICRO INDUSTRIES	75	40	13	•		•	•	•
MINT CORP.	60	54	4	•		•	•	•
NATIONAL TELEPHONE & ELECTRONICS INC.	50	12	22	•		•	•	•
PLEXUS	1350	270	12	•		•	•	•
PREGITZER INDUSTRIES	108	45	7	•		•	•	•
PRECISION GRAPHICS INC.	36	12	21	•	•	•	•	•
R&M ELECTONICS INC.	60	27	13	•		•	•	•
RAMP INDUSTRIES INC.	200	150	16			•	•	•
SEAFAB INC.	50	11	14			•	•	•
SOLECTRON CORP.	2000	500	15	•		•	•	•
TURTLE MOUNTIN CORP.	165	47	18			•	•	•
WESTERN RESERVE ELECTRONICS INC.	100	35	32	•			•	•

7.2.1 Quality/Reliability Plan

Maximum price leverage is gained by establishing dual use facilities serving both commercial and DoD customers. The key to meeting military standards in this environment is qualifying the process and not the product. This can be accomplished through a Qualified Manufacturing Line (QML). We believe a QML facility will best serve both commercial and DoD customers. The facility will be driven by principles of Total Quality Management. These benchmarks of excellence are the standards represented by the Malcom Baldrige National Quality Award.

Process improvements will be continuously inserted as part of the continuous improvement process. Process improvements will be baselined using test element groups. Manufacturing coupons will travel through the manufacturing process concurrent with deliverable elements.

The QML qualification vehicle will be a standard evaluation circuit specifically designed for that purpose.

7.2.2 Production Test Needs

Today test capabilities are not evenly distributed across the technologies (Figure 55); however, the requirements are similar. All elements of the system require functional verification and performance qualification. The most mature and robust capabilities may be found in the PWB assembly and fabrication technologies. As we follow the silicon trail back to its source we find less mature capabilities in place.

7.2.3 Test Solutions

No single test solution is complete by itself. The total solution is made up of capable hardware, test methods, mechanical interfaces equipped with controllability and observability, and standard I/O.

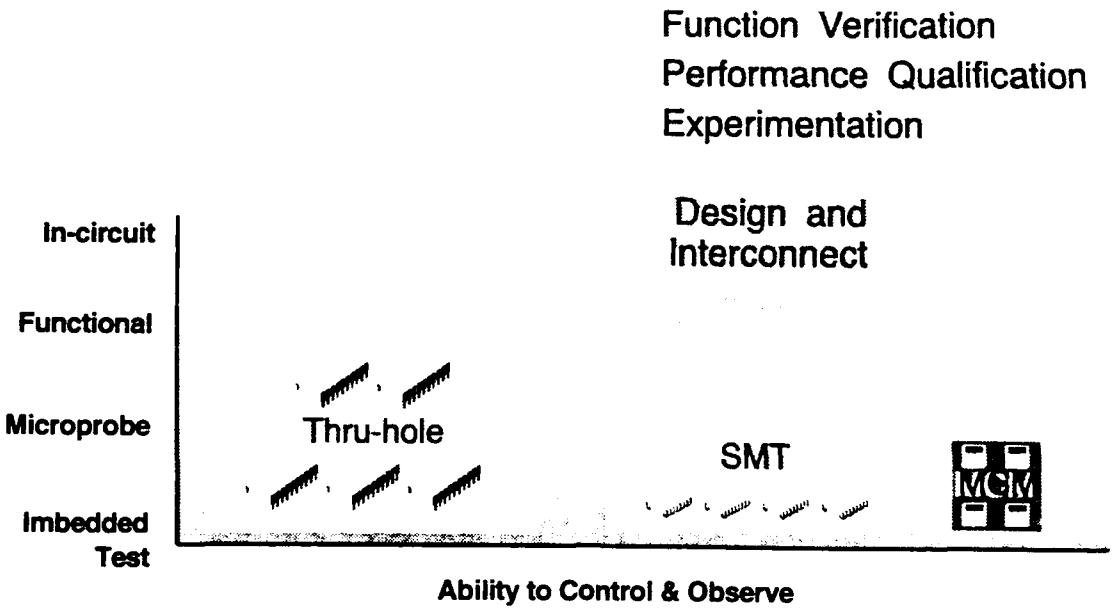


Figure 55
Production Test Needs

7.2.4 Hardware Foundries - Test Interface

The long term success for test is tied to self-test. As a minimum, near extension to EDIF and WAVES is required to support existing VHDL vector and net-list format communication. Boundary scan format standards are being worked and proposed. To take advantage of these standards the CAD vendors must output design representations in these formats to allow future working tools to function optimally. We must also determine what feedback data is most appropriate and in what format it can be provided from test to the design systems to aid in future design decisions.

7.2.5 Test Technology Roadmap

Figure 56 depicts three separate roadmaps for MCM test. This chart depicts the emergence of built-in-test which is expected to evolve as the standard test methodology of choice. In addition we see manufacturing defects test and performance test merging in the mid-to-late 1990's.

The near term will be dominated by interim solutions as emerging technologies demand test capabilities not performed. "Known good die" is a key capability for MCM technologies. The MCM manufacturer cannot efficiently test die outside the wafer format. Test at this level subjects the die to handling damage and is a very costly process. The silicon manufacturers, must step forward to fill this requirement. TI, as a silicon manufacturer, is participating in infrastructure activities in this area and has pledged to provide this capability.

7.2.6 Manufacturing Control

As illustrated in Figure 57, our foundries are equipped with a variety of CIM systems. These systems have emerged from islands of automation and from legacy system architectures.

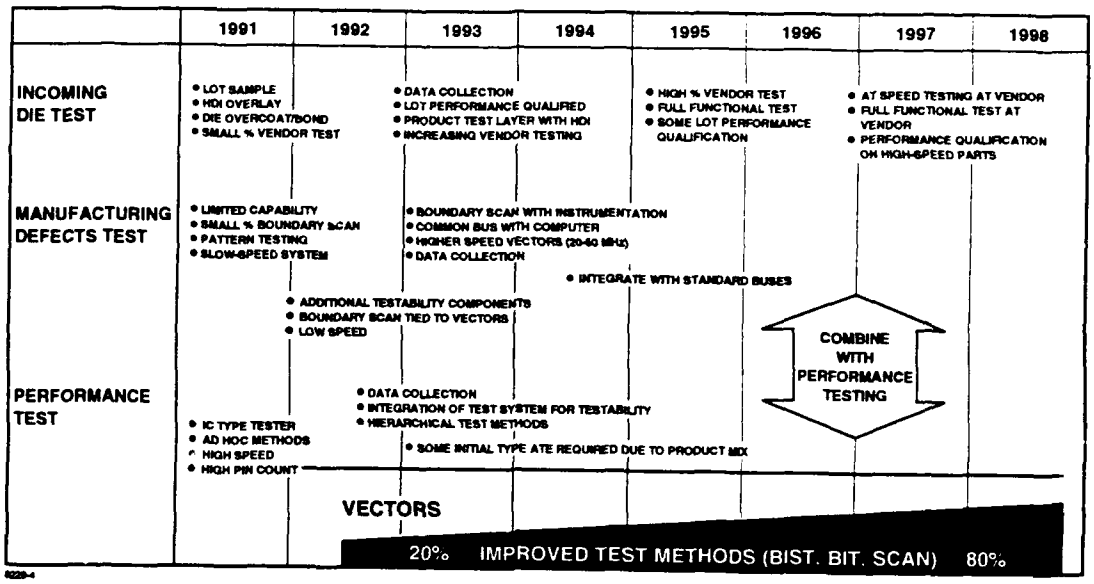
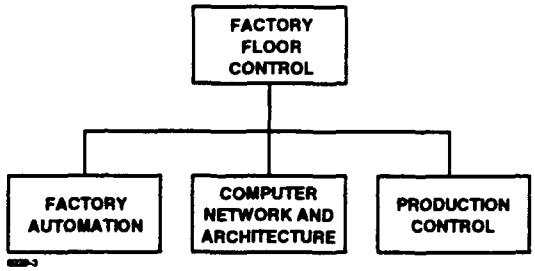


Figure 56
Test Technology Roadmap



CAPABILITIES:

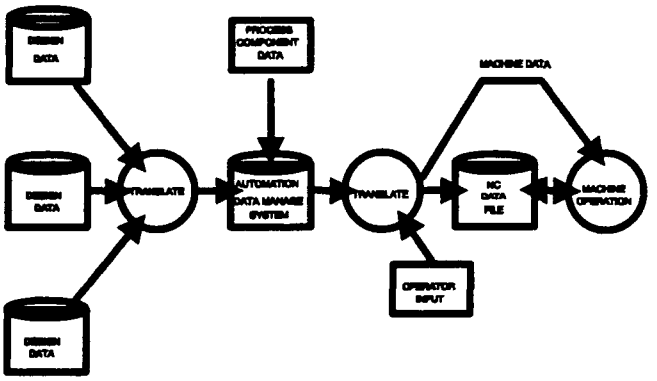
- Factory automation systems tailored to dissimilar equipment control requirements
- Custom design system interfaces
- Production control systems providing local work-in-process and process control information

BARRIERS:

- Unique equipment interfaces prohibit development of 'brilliant' equipment capabilities required to support RASSP objectives
- Proprietary design system output formats require the development of multiple technology specific translators
- Inability to entend production modeling capabilities and status feedback to customers and suppliers
- Portability of system capability limited to specific platforms

Figure 57
Foundry Manufacturing Control

We find automation systems, shown in Figure 58, which have been custom engineered around a lack of standards with many custom interfaces. Unique data translators have been constructed to import various design system outputs. Many machine do not have automated interfaces and must be manually programmed or taught. We expect to find a future environment in which equipment will contain "brilliant" capabilities. The foundry CIM system must be architected to communicate with this type of equipment. This means the equipment must be given a "peer" role in the CIM system to take advantage of the local decision making capability of this new equipment type.



Capabilities:

- O Unique translators to import proprietary design data bases.
- O Proprietary data management systems.
- O Manual programming or unique data translators for each machine.

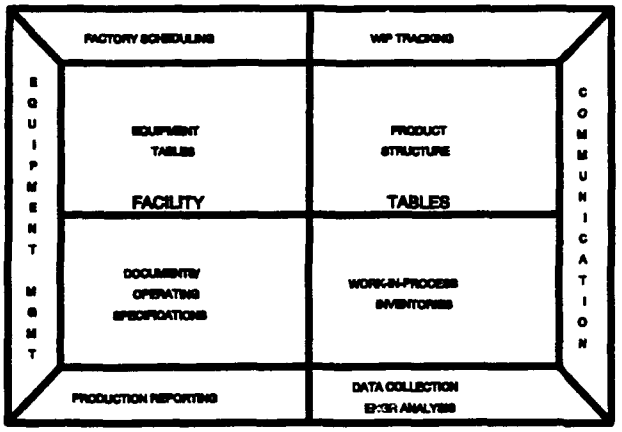
Barriers:

- O Lack of standard product descriptions.
- O Commercially available data generation capabilities based on common product definition standards.
- O Lack of integrated equipment automation and production control - equipment not viewed as a 'peer' in the system architecture.
- O Lack of coordinated product data management capabilities.

Figure 58
Foundry Data Automation System

Design systems of today produce a variety of outputs which must be dealt with by the foundries. In an environment of no standardization the foundries can best defend themselves by not trying to extend these "non-standards", but place rules on the construction of the designs so the outputs can at least be interpreted.

Production control systems, Figure 59, have been designed for use by foundry personnel primarily for the purpose of controlling the vast amount of work-in-process on the factory floor and providing limited process feedback to operators. Customers and suppliers have had a limited ability to participate in these capabilities. In the future the foundries will have to extend the capabilities to both suppliers and customers in an emerging environment of teaming. Suppliers will be expected to monitor materials use and plan shipments of raw materials to arrive just-in-time to support manufacturing operations. Customers will monitor progress of their orders in the factory, helping to set schedules and priorities tied to their changing needs. The systems of today have limited simulation capability and only loose connectivity to automated equipment operations.



Capabilities:

O Local production control management.

O Work-in-process management.

O Production dispatching, scheduling, alarms, ...

Barriers:

O Customer / supplier access limited.

O Lack of integrated process simulation tools.

O Loose connectivity to automated equipment control functions.

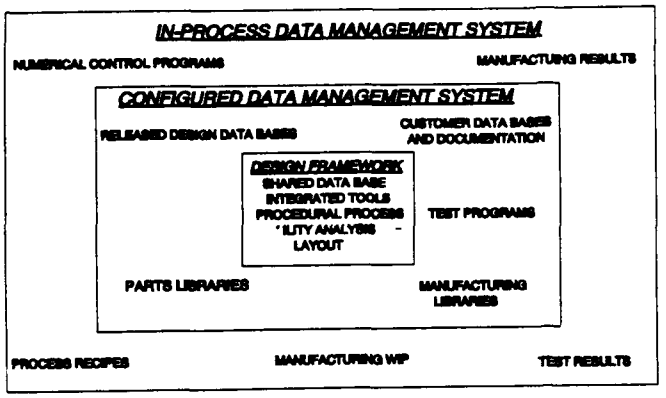
Figure 59
Foundry Production Control System

Our view a Foundry Data Management systems consists of the layered structure shown in Figure 60. The kernel consists of design system frameworks such as the Mentor Falcon framework. These frameworks integrate design tools around a common data base in a concurrent engineering environment and provide for local configuration management of data within the data base.

The next layer in the structure is secured and formally configured data. This layer contains formally managed data such as released designs, customer supplied data bases, part libraries, and manufacturing libraries. Data in this structure is formally released and controlled through formal configuration management procedures and structures. This layer can be supplied through commercially available products such as SDRCs DMS, and Sherpa. The final layer consists primarily of in-process data. Access to this data requires ad-hoc methods and queries. Typical data in this layer are manufacturing results, manufacturing WIP status, process recipes, NC programs, etc. This data may reside in custom foundry data bases or within commercial factory control application data bases.

7.2.7 The Future of Computer Integrated Manufacturing

Future systems will be required to be portable, distributed, and scalable to support both low and high volume requirements. The system will be expected to contain embedded decision making capabilities, able to process the large amounts of real time data being continuously updated within the system. The decisions will optimize the factory throughput to maximize the ability to meet customer need dates.



Capabilities:

- O Design frameworks to integrate design tools around a common data base in a concurrent engineering environment, providing local configuration management.
- O Product data management system to provide secured access and formal configuration management of released product description files.
- O In-process data management system allowing ad-hoc access to in-process data facilitating analysis and feedback.

Barriers:

- O Customer access and particiaption in product data management system.

Figure 60
Foundry Data Management

Product specification and simulation capability will provide a process synthesis capability to help validate the partitioning decisions, and aid in the selection of a foundry with capabilities that best match the specific design needs. For instance, the design advisor may partition a potion of the system to be manufactured in a MCM, but which technology? Flip-chip, Tab, Overlay? The process synthesizer will help answer these questions. These capabilities must also be provided in a seamless environment to both the customer and supplier.

7.2.8 Next Generation Manufacturing System Contacts

TI has been working with several suppliers of manufacturing execution systems (MES), Table 22. It is our objective to form a strategic relationship with one of these suppliers to provide the next generation system to meet the needs of our advanced manufacturing technologies.

We have had recent discussions with both Promis™ and Consilium because both of these systems are currently installed in existing defense systems facilities. We have determined that both of these suppliers have plans to move toward open architectures, GUIs, and RDB. OODBs are also in the future possibilities for both companies.

TI has previously submitted a DARPA ASEM proposal in which current capabilities of the MMST system would be extended to cover additional requirements of multichip manufacturing.

7.2.9 Next Generation Manufacturing Issues

Table 23 identifies several key issues which effect the objectives of the RASSP program. We believe that for the most part these needs are

Table 22 Next Generation Manufacturing System Contacts

NEXT GENERATION MANUFACTURING SYSTEM CONTACTS

COMPANY	PRODUCT	RASSP APPLIC	TIME FRAME	TI POC	COMPANY POC	COMMENT
Promis systems	PROMIS	Mfg control	95	D. Counts	P. Jones	Unix GUI Automation I/F
Consillium	Workstream	Mfg control	95	D. Counts	TBA	Unix RDB
Texas Instr.	MMST	Mfg control	97	D. Counts	J. Mahaffey	Unix OODB GEM++ Substantial functionality step function improvement.

Table 23 Next Generation Manufacturing Issues

ISSUE	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Lower cost, manufacturing capability, cycle time	Process improvement/technology	Yes	Yes	DARPA foundry programs DARPA ASEM program Mantech Foundry commercialization	1993-1995	Continued support of key technology contributors
Process modeling and simulation	Embed tools in next generation CIM system	Yes	Yes	MMST Program DARPA ASEM program	1996-1997	Extend capability to include key RASSP technologies
Customer access to foundry	Standard product descriptions and communicabons infrastructure	Yes	Yes	DARPA ASEM program Standards Committee(s)/MCC	1996-1997	DARPA sponsorship of efforts key to RASSP
Continuous improvement	Closed loop manufacturing control	Yes	Yes	MMST program	1996-1997	Continued DARPA sponsorship of technology improvements
Translation of test from customer	Standardized test vector format, integrated with standard diagnostics, on high speed test equipment	Yes	Yes	Industry starting to address	1993-1994	Support of industry standard vector format
Generation of manufacturing defects tests for MCM's	Automatic test pattern generation with & w/o boundary scan	Yes	No	Extension of current commercial software		Sponsor enhancement of commercial product
High pin count, high speed probes are expensive $8K $10K	Process improvement development	Yes	No	Commercial probe vendors		Sponsor enhancement to assembly process and probe development

being addressed by existing programs and internal commercialization plans. It is our recommendation that these programs be continued to enable the objective of the RASSP program to be achieved.

8. BUSINESS PLAN AND PROGRAMATICS

The following section describes approaches to the development and management of the RASSP program. First, the section discusses the business infrastructure and teaming environment required for RASSP to become successful. Second, a potential program plan is described. This program plan will bring together the critical organizations required to put RASSP in place in the DoD signal processing community. Finally, the section discusses several potential applications that could be used for the final RASSP demonstration. Significant pieces of these applications can be used as interim benchmarks of RASSP capabilities, providing continual measurement of progress against the RASSP goals.

8.1 Business Infrastructure and Teaming Environment

RASSP is a highly interdisciplinary activity integrating advanced technology, as shown in Figure 61. The expertise required to develop the RASSP concept spans many organizational boundaries. RASSP technology must flow rapidly across these boundaries to achieve RASSP's cycle time goals.

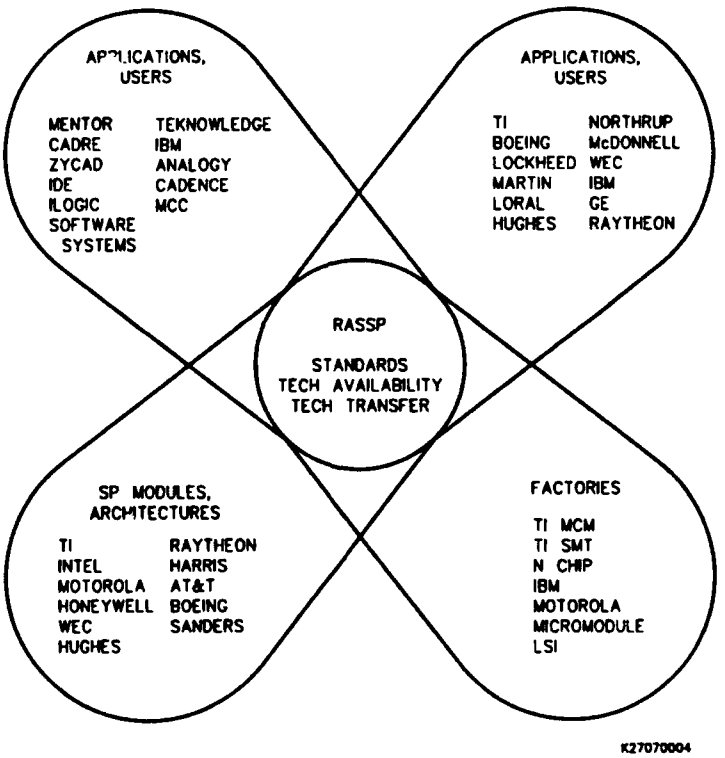


Figure 61
Teaming

RASSP development roughly can be partitioned into four organizational segments, corresponding to the evolving signal processing business segments we see today. The segmentation between organizations is not pure, and is evolving. However, these four key elements are all necessary contributors to a RASSP program.

The corporations listed in Figure 61 are merely examples of companies that compete in the four RASSP segments. This is by no means a complete list. The business pursuits and fortunes of the many potential companies evolve, sometimes quite rapidly. Many of the leaders in

today's tool market were not major players as little as five years ago. This dynamic trend likely will continue. RASSP must develop and extend a business infrastructure that transcends the dynamic nature of the business today.

Standards, technology availability, and technology transfer are critical elements of this business view. When RASSP is mature, perhaps in the year 2000, these will be inherently available through the normal business infrastructure in a manner similar to the infrastructure of today's ASIC business. However, during the evolution of RASSP, DoD and industry must focus on these critical elements that enable the RASSP process.

The ideal RASSP program will pull together many organizations from the four segments and focus their activities on the common RASSP goal. This could evolve from the normal business relationships established today. However, this will not happen fast enough to meet the RASSP needs. Instead, DoD must provide this focus through RASSP demonstration contracts. These contracts will promote and fund the interdisciplinary activities of RASSP toward specific RASSP demonstration goals.

Teaming between the numerous organizations will be required. Most of the major corporations in each of the critical areas must be involved, in some way. DoD must manage, promote, and fund the RASSP infrastructure to insure the maximum availability of RASSP technology to industry and the DoD.

8.2 Program Plan

Figure 62 shows a possible RASSP program plan. This plan consists of four elements:

- Standards Consortium
- Technology Program
- Tools and Library Program
- Final Demonstration.

8.2.1 Standards Consortium

RASSP is an interdisciplinary process and infrastructure, not a product. Industry supported and open data representation and data exchange standards and hardware and software interface standards are critical to rapid development and deployment of the RASSP process and infrastructure. Through open standards, all of the organizations and corporations engaged in the four RASSP segments ultimately can be included in RASSP. Proprietary and competitive tools, library elements, or manufacturing capabilities easily can be included as they evolve.

Standards are a prime and fundamental issue. DARPA should sponsor a RASSP standards group or consortium activity to select standards and infrastructure for initial RASSP demonstrations.

The RASSP standards consortia will organize into several sub-working groups and/or committees in the critical areas to select the data base, data exchange, software and hardware standards for RASSP demonstrations. The charter of this consortium will be to select from among existing standards, not to develop new ones. The consortium must act quickly to support the first RASSP demonstrations. Initial standards must be in place no later than 12 months into the program, with a final freeze on demonstration standards at 18 months. However, standards will evolve with RASSP. The consortium must continue to influence and select standards throughout the RASSP program.

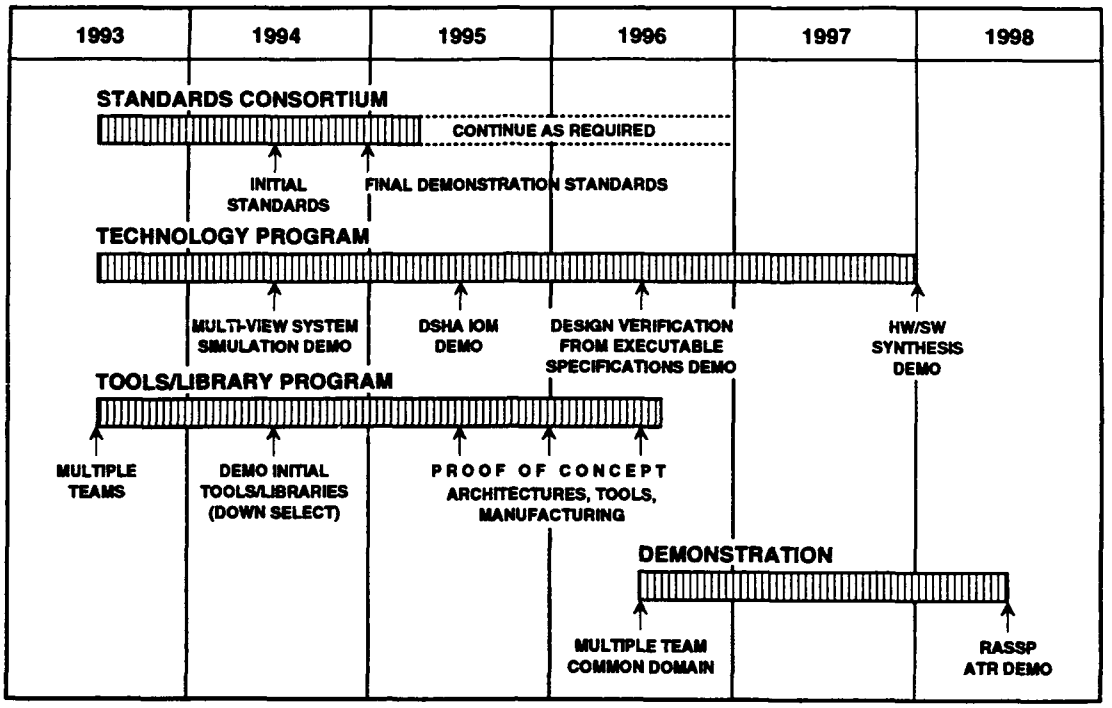


Figure 62
Program Plan

The standards consortium can be organized according to the 1984 National Research Act, which enables firms to engage in pre-competitive research without breaking antitrust laws. The consortium should include systems houses, processor suppliers, merchant suppliers, software developers, universities, and government agencies and laboratories. The NCMS, addressing the manufacturing objective of affordable defense systems, is a good template for the RASSP standards consortium.

8.2.2 Technology Program

Many excellent ideas and technologies are emerging with the potential for significant impact on RASSP. These need to be promoted and focused on RASSP specific issues through a RASSP technology program. The highest impact areas for this technology program are in technologies for system design, analysis, and synthesis using modular, library elements. The initial emphasis should be on multi-view (performance, function, cost, reliability, etc.) system simulation and analysis, migrating to design verification and requirements verification technologies, and finally concentrating on high-level hardware and software partitioning and synthesis of systems from large-grained modules.

8.2.3 Tools and Library Program

A domain specific RASSP library and library design tools program is central to achieving the needed design cycle times for RASSP. This program would be a 36 month program that: (1) develops application specific hardware and software library elements, (2) integrates and uses

available design tools to implement the library out-cycle designs, (3) develops and incorporates application specific, in-cycle design tools, and (4) integrates rapid design implementation into the design flow through foundries and ASEM capabilities. Program progress will be monitored with "proof of concept" demonstrations that benchmark the in-cycle cycle time of pieces of the design process using the individual tools and library elements as they are developed and integrated into the system. At least one demonstration will be carried through to implementation in a ASEM foundry or foundries to benchmark the foundry interface.

The tools and library program could incorporate multiple awards with multiple teams. Selection emphasis should be made on the availability, flexibility, and ease-of-use of the resulting library and tools in the in-cycle design and implementation demonstration that will follow. First-time, out-of-cycle design capabilities and tools for library elements should not be a major selection criteria.

The tools and libraries team(s) will include members from each of the four RASSP segments. Each team profile will include a system house(s) to manage the vision and domain requirements, as well as the key library and process elements provided by the CAD/CASE/CAE vendors, signal processing and architecture vendors, and the fabrication foundries.

8.2.4 Demonstration

The RASSP program will conclude with a demonstration of RASSP in-cycle capabilities using the tools and libraries developed in the tools and library program element above. The demonstration program should be of 24 month duration to allow time for initial independent benchmarking of the tools and libraries and some out-of-cycle augmentation of the libraries with application specific and/or proprietary library elements. The actual in-cycle design and demonstrations should be started and completed within the last 12 months of the demonstration program. The program should demonstrate the design of a complete signal processing system, from system design through integration and test.

Multiple demonstrations should be encouraged. The demonstration program provides RASSP technology transfer within the user community. The domain should be limited to Automatic Target Recognition, or another suitable limited application domain to provide maximum domain specific focus for library and tool development.

8.3 Applications

Perceived RASSP success depends on the selection of the final demonstration applications. The demonstration must clearly and convincingly show the tremendous in-cycle cycle-time improvements resulting from the RASSP methodology on a mainstream application.

8.3.1 Applications Criteria

Several criteria must be considered in selecting a RASSP demonstration application:

Important Application: The application must have a high value and visibility within the Department of defense.

Funding Available: Funding must be available to carry on from the demonstration into a fielded application.

Suitable Software Content: Software is a critical element of modern signal processing applications. The application must have a

sophisticated software requirement typical of modern weapons systems. An application with less that 20 thousand lines of code would not be considered representative of typical applications today, while an application with more than 200 thousand lines of code would require a major software development effort that would detract from the key focus of the demonstration.

Clear Computational Model: A clearly understood functional model of the candidate application should be available at the outset of the demonstration. This allows the demonstration to focus on the key design elements of RASSP rather than on trying to figure out algorithms to solve the problem. Algorithm invention is beyond the scope of the RASSP program. An application with a clear computational model best matches the model year upgrade scenario for RASSP.

Challenging Computational Performance: Most modern weapons systems demand significantly more throughput than commercial systems. An application should require greater than 1 billion operations-per-second performance to be typical of today's real-time signal processing systems. An application with greater than 10 billion operations-per-second performance is a sufficiently large and complex hardware effort that it would detract from the systems design and software activities.

Extended Life Cycle: The application should have a reasonably long lifetime with expected upgrades and improvements. The design cycle time improvements for RASSP are best exploited in systems that require rapid "model year" upgrades.

Multi-platform Applications: Applications that have the potential for use on multiple platforms demonstrate the ability of RASSP to adapt rapidly to differing environmental, conformal, or logistics requirements.

Prototype Implementation Schedule: The application implementation schedule must be consistent with the RASSP program schedule. The RASSP schedule should not be perceived as a limiting factor to deployment.

During the course of this RASSP study, TI and Boeing evaluated several potential RASSP demonstration applications using the criteria outlined above. From this evaluation, five excellent applications were identified that are ideally suited for demonstration of RASSP. These are:

- TACAWS Seeker (Army) — Precision strike anti-armor, anti-helicopter missile
- Noncooperative Target Identification (Air Force and Navy) — Location and identification of targets from combat aircraft
- E-3 AWACS Infrared Search and Track (Air Force) — Detection and tracking of targets and cruise missiles from standoff surveillance platform
- Automatic Ship Classification from Inverse Synthetic Aperture RADAR (Navy) — Ocean surveillance and shallow water anti-submarine warfare
- Acoustic Intercept System (Navy) — Multi-platform anti-submarine warfare.

These applications all fit well within the criteria described. In particular, these applications share three common characteristics that make them well suited to RASSP demonstration. These are:

- Suitable software content and throughput requirements

- Well understood computational models
- Rapid upgrades mandated by new sensors, sensor upgrades, and new threats.

8.3.2 Application Evaluation

8.3.2.1 TI Applications Criteria. Tables 24 - 27 provide detailed characteristics of applications evaluated by TI for potential RASSP demonstrations. The applications include systems under development both inside TI and at E-Systems.

8.3.2.2 Boeing Applications Criteria. Tables 28 through 31, and Figures 63 through 70 describe characteristics of several potential applications studied by Boeing during the RASSP Study Phase. Selection criteria are the same as those provided for the Texas Instruments programs. Key for column entries in the carts as shown by the table Application Selection Criteria for a RASSP Demonstration System.

Tables 29 and 30 that follow list the potential RASSP applications and associated scoring. Applications are grouped by category. Categories are Precision Strike, Missile Defense, Navy Undersea, and Navy Airborne Early Warning (AEW) system. With each program we provide the scoring values, contacts at Boeing, occasionally contacts in the government, and selected comments mostly related to requirements.

Some of the programs listed in the tables would make particularly attractive demonstration systems for RASSP. The addition of infrared sensor capability to the radar surveillance aircraft Navy AEW system would require new image processing electronics. This is a long life program and we would anticipate model year upgrades over the life of the program. Further, the signal processing challenge of integrating radar and IR data aboard the surveillance/battle management platform will provide high visibility for the benefits of RASSP.

The Acoustic Intercept System is a new system under development by the Navy. Significant signal processing will be required for this system. Program schedules should match the RASSP schedule quite well. Further, the ability to move with the model year concept would be a key demonstration element if this AIS system were chosen for demonstration.

Included in the candidate program list are interceptor and seeker electronics. Several programs in this category are also listed by Texas Instruments. Also, there is the potential for significant performance improvement in new generation space systems. Brilliant Eyes, and the Future Early Warning System (FEWS) are two examples. However, the space community can be very conservative when considering use of new technology, and there may not be sufficient schedule leeway for use of RASSP. However, the benefits would be significant.

In addition to the programs listed in the tables, there are several carrier and platform vehicles that offer the potential for important RASSP application demonstrations. The platform systems generally have long lives - more than three decades for some systems. During their lifetime, these programs undergo many electronics upgrade cycles. RASSP offers the potential for very major cost savings by using the model year concept for such upgrades.

Table 24 TI Potential RASSP Application

Application	TACAWS Seeker	Noncooperative Target ID	JSOW/JDAM Seeker	Program A
Mission	Precision Anti-Armor, Anti-Helicopter Missile	Aircraft Precision Strike, Air Systems/Defense (Classified)	Air-Ground Seeker	Air-Air Weapon
Service	Army	Air Force, Navy	Air Force, Navy	Classified
Importance	JAWS Seeker Technology	Multi-Platform, AX, MRF	Precision Strike	
Funding Probability	75%	85%	90%	65%
Lines Of Code	30-70K	20-40K	60K	150K
Computational Model	Baseline well understood	Well understood	Complete	90% Complete
Throughput	0.5-1.5 GFLOPS	1.0-3.0 GFLOPS	30 MFLOPS	1.5-2 GFLOPS
Implementation Stage	MMS captive flight 93/94	ATD	ATD	ATD
Platforms	Missile	Combat Aircraft, UAV	Multiple Missiles	
Demo Schedule	1996	1996	1994 EMD	1997 EMD
TI Contact	Roger Elkins (214) 462-4421	George Niemann (214) 480-2417	Ken Martindale (214) 462-4209	John Summerford (214) 995-6923
DoD Contact	Dr. James Bradas MICOM	See George Niemann	See Ken Martindale	See John Summerford
Comments	Excellent Demo Candidate	Excellent Demo Candidate	Design well established	Good candidate

Table 25 TI Potential RASSP Application

Application	Program B	Program C	Program D	Program E
Mission	Target Acquisition Airborne Sensor	ATR, Air-Ground Weapon	Navigation, Air-Ground Weapon	Air-Ground Weapon
Service	Classified	Classified	Classified	Classified
Importance				
Funding Probability	95%	95%		65%
Lines Of Code	20K	70K	33K	130K
Computational Model	Complete	Complete	Complete	60% Complete
Throughput	2-5 MIPS	25 MFLOPS	4 MFLOPS	30-60 MIPS
Implementation Stage	Production	EMD	EMD	EMD
Platforms	Multiple	Multiple		
Demo Schedule	Production	Production 95	Production 95	Production 95
TI Contact	John Summerford (214) 995-6923	John Summerford (214) 995-6923	John Summerford (214) 995-6923	John Summerford (214) 995-6923
DoD Contact	See John Summerford	See John Summerford	See John Summerford	See John Summerford
Comments	Too late to influence	Too late to influence	Too late to influence	Software intensive

Table 26 TI Potential RASSP Application

Application	Automatic Ship Classification from Inverse Synthetic Aperture Radar	F-117 Video Tracker	AAQ-17 Video Tracker	LAMPS
Mission	Ocean Surveillance (ASUW) and shallow water ASW	Precision Guided Weapon Delivery	Fire Control	Surveillance
Service	Navy, Coast Guard	Air Force	Air Force	Air Force
Importance	Operational requirements: target identification, surface ship classification, periscope detection and discrimination	Update to existing system	Update to existing system	Update to existing system
Funding Probability	85%	30%	70%	70%
Lines Of Code	30-60K	20K	20K	50K
Computational Model	Well understood	Well understood	Well understood	Mission dependent
Throughput	1.5-2 GFLOPS	660 MFLOPS	660 MFLOPS	660 MFLOPS
Implementation Stage	ATD, lab and near real-time	Operational	Operational	EMD
Platforms	S-3, P-3, USCG C-130, LAMPS MK III, USCG Cutters, USN Combatants	F-117	AC-130	HH-60
Demo Schedule	Shore 1995, airborne 1996	1996	1992 96-97 Production	1992 96-97 Productions
TI Contact	Bob Huffer (214) 952-4907	Dale McCutcheon (214) 480-6460	Charles Grassl (214) 480-6847	Charles Grassl (214) 480-6847
DoD Contact	See Bob Huffer	WRAFP see Dale McCutcheon	Warner Robbins see Charles Grassl	NAVAIR see Charles Grassl
Comments	Excellent candidate, potential for early EMD	Upgrade from hard-wired to DSP solution	Upgrade from hard-wired to DSP solution	More extensive software

Table 27 TI Potential RASSP Application

Application	AX Video Tracker	Digital Signal Processor
Mission	Precision Guided Weapon Delivery	SIGINT collection and processing of spread spectrum and frequency hopping signals
Service	Navy	
Importance	Critical Capability	Next generation airborne reconnaissance
Funding Probability	50%	80%
Lines of Code	200K	20-50K
Computational Model	Mission Dependent	Well understood
Throughput	10 GFlops	4-6 GFlops
Implementation Stage	Dem/Val	Commercial, board level implementation
Platforms	AX	Multiple, classified
Demo Schedule	1997	See Mark Wilhelm
TI Contact	Bob Stewart (214) 480-6311	Mark Wilhelm (E-Systems) (903) 457-4521
DoD Contact	Jesse Wijntjes, NAVAIR	See Mark Wilhelm
Comments	Extensive Software	Good Candidate, with P^3I plan for an existing system

Table 28 Boeing Application Selection Criteria for a RASSP Demonstration System

Column Key	Item
1	Important application
2	Funding available
3	Application code requirements are appropriate
4	Clear computational model
5	Challenging computation performance
6	Life-cycle correct for new or existing system
7	Multi-platform application
8	Prototype implementation schedule

Table 29 Boeing Potential RASSP Application - Scoring

Category	Application	RASSP application selection criteria								Contacts	Comments
		1	2	3	4	5	6	7	8		
Precision strike	Airborne surveillance platform—IR/radar fusion	8	3	6	5	8	7	9	5	Air Force: _____ Boeing: Bloedel	Multiple processors required
	Airborne missile interceptor launch platform (airborne) signal processor suite	4	2	6	3	5	3	6	6	Air Force: _____ Boeing: Bohrer	Requires systems approach
	Postboost high-end/exo LEAP seeker signal processor	8	5	7	7	8	5	7	7	Navy: _____ Boeing: Norsworthy	May require satellite/interceptor interaction
Missile defense	Ground-based interceptor	10	10	8	7	6	9	2	1	Army: _____ Boeing: Donlin	Processors may be specified
	Surveillance satellites BE	5	7	3	3	5	3	1	4	Air Force: _____ Boeing: Willard	
	FEWS	10	7	5	6	4	6	1	6	Air Force: _____ Boeing: Willard	Excellent potential RASSP
	Command center processing	10	4	5	4	4	4	3	7	Air Force: _____ Boeing: Stamper	Requirements still in development

Scoring:
10 Excellent fit for RASSP application
1 Poor fit for RASSP application

Table 30 Boeing Potential RASSP Application - Scoring

Category	Application	RASSP application selection criteria								Contacts	Comments
		1	2	3	4	5	6	7	8		
Navy – undersea	Acoustic intercept system	7	0.5	20K	6	2G	7	10	10	Navy: _____ Boeing: Miller	Excellent RASSP potential
	SOMSS: mine-avoidance sonar	6	0.5	40K	5	3G	7	4	8	Navy: _____ Boeing: Sinfield	–
Navy – AEW	E-X: next-generation aeroborne early warning (carrier-based aircraft)	6	0.4	80K	7	10 G	8	2	5	Navy: _____ Boeing: Gilbert	Stressing application

Table 31 lists airborne platforms for potential demonstration. The chart lists first several programs that are already in operational service. As indicated, upgrades are an important part of such programs. Development aircraft are also listed. Opportunities exist for using RASSP signal processors in such programs. However, in the development stage, electronics suppliers and design procedures are generally well established. New programs are also listed in the applications tables. New programs offer the most flexibility in design and application of RASSP procedures. However, program funding is not so certain as in the development programs. The E-X, next generation Navy carrier-based airborne warning aircraft offer an especially important opportunity for RASSP application. Signal processing will be stressing for E-X. The platforms for potential are shown in the next chart.

Table 31 Additional Airborne Platforms for RASSP Application

Vehicle	Category	Status	Comments
B-2	Bomber	Existing	Excellent RASSP potential
B-1B	Bomber	Existing	–
B-52G/H	Bomber	Existing	Continuing upgrades
E-4B	Airborne command post	Existing	Continuing upgrades
F-22	Fighter	Development	Excellent RASSP potential
LH	Attack helicopter	Development	–
V-22	Tilt-rotor carrier	Development	–
A-X	Fighter	Proposed	–
MRF	Fighter	Proposed	–
E-X	Airborne early warning	Proposed	Excellent RASSP potential

8.3.2.3 Processing Flow for Selected Applications. Typical processing flow for optical surveillance consists of detection electronics to interface with optical sensors, analog electronics for signal conditioning and preprocessing, digital signal processing for both time-dependent and object-dependent processing, and data processing for mission-dependent computations. A framework for this processing sequence is shown in Figure 63.

Within the structure shown, there are several basic functional blocks. These blocks provide the opportunity to establish some basic interfaces. For example, basic interface mechanisms can be established at the sensor/signal processor interface, at the signal processor/data processor interface, and at various points in between. Similarly, interfaces for other application of signal processors can be established. For example, the following illustrations (Figure 64 - 68) show a generic processor, typical electro/optical sensor signal processors for space surveillance, a space-based radar processor, synthetic aperture array processor, and an electronic support measure processor. In addition to interfaces, there are needs for handling control of the processing elements in a more standardized way. Control conventions can help establish the model year upgrade process, just as can establishing data handling conventions.

There are many functional elements common to all these processors. Requirements, algorithms, designs, software, test cases, and related documentation for commonly occurring elements can and should be available to developers in a RASSP library. The library should contain hardware, or fixed logic, designs as well as software solutions for implementation.

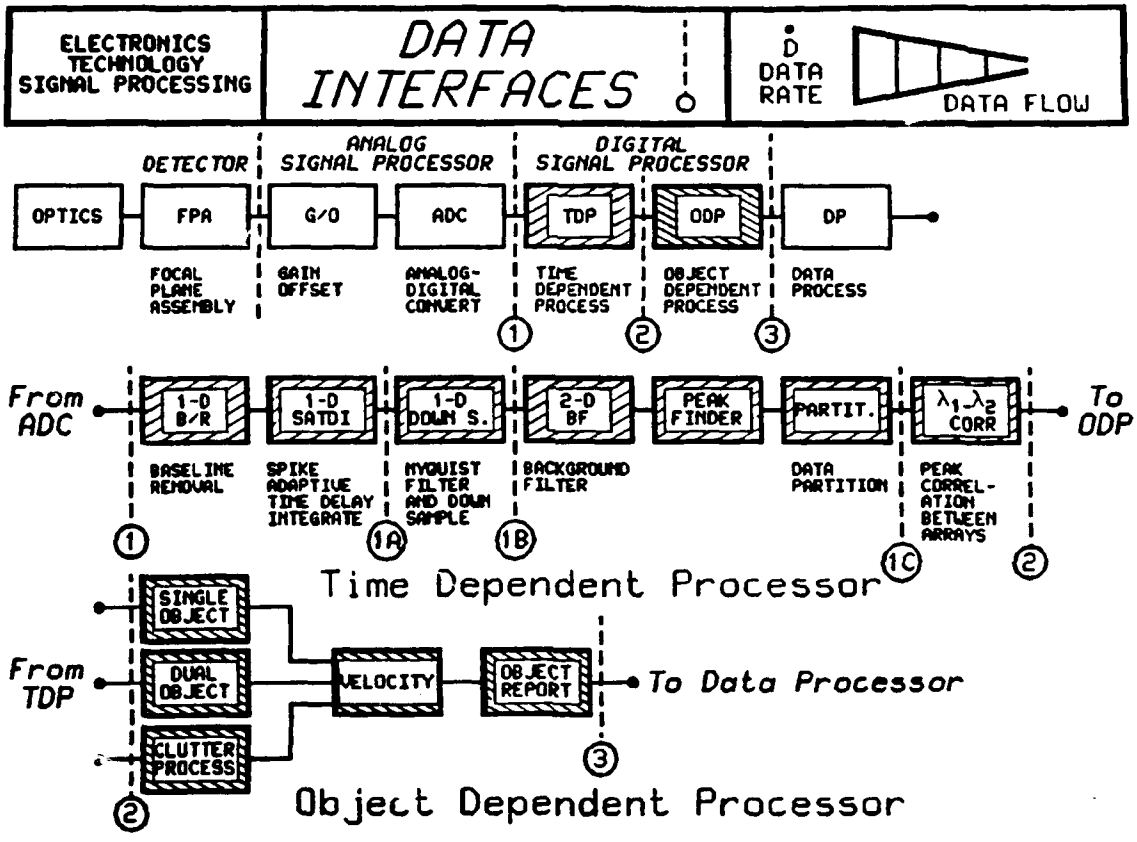


Figure 63
Processing Flow

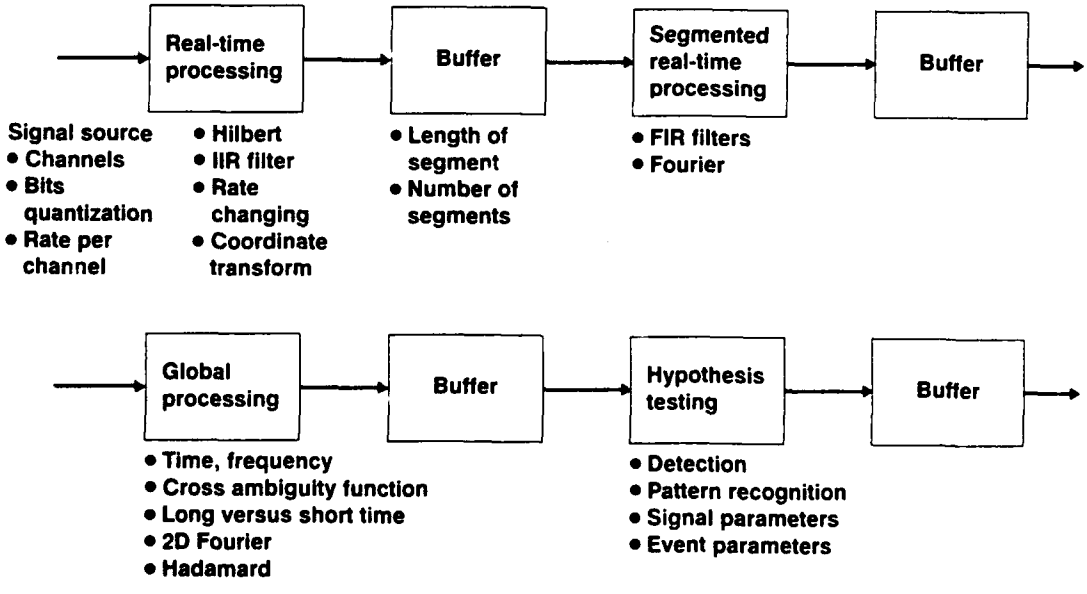


Figure 64
Typical Scanning E/O Sensor (Space Surveillance)

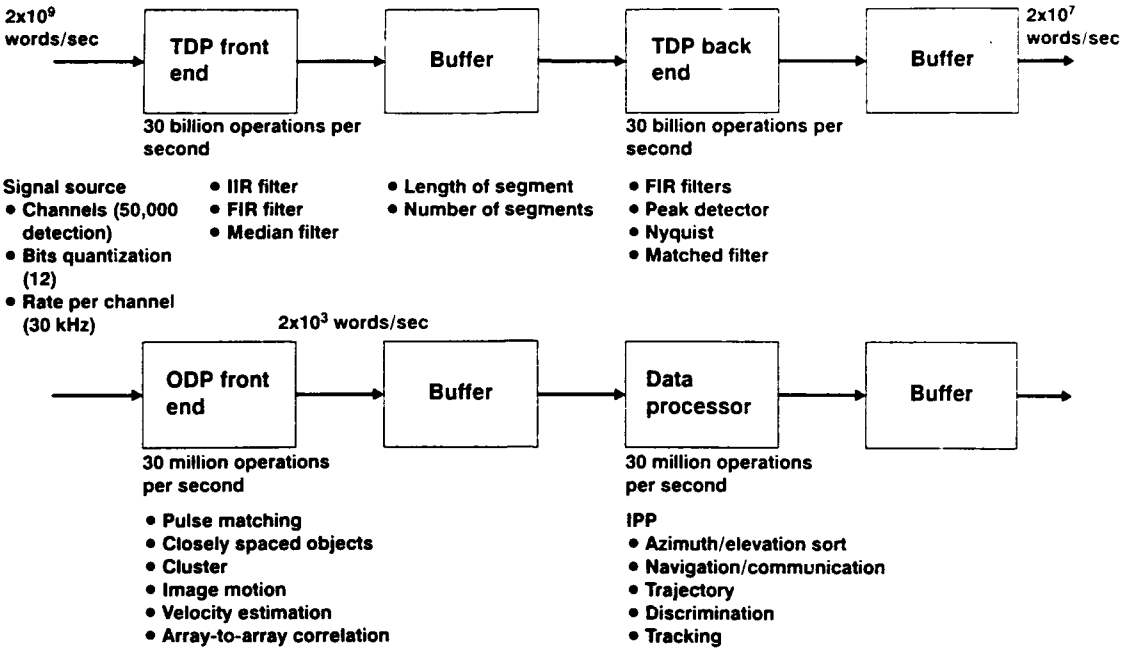


Figure 65
Generic Processor

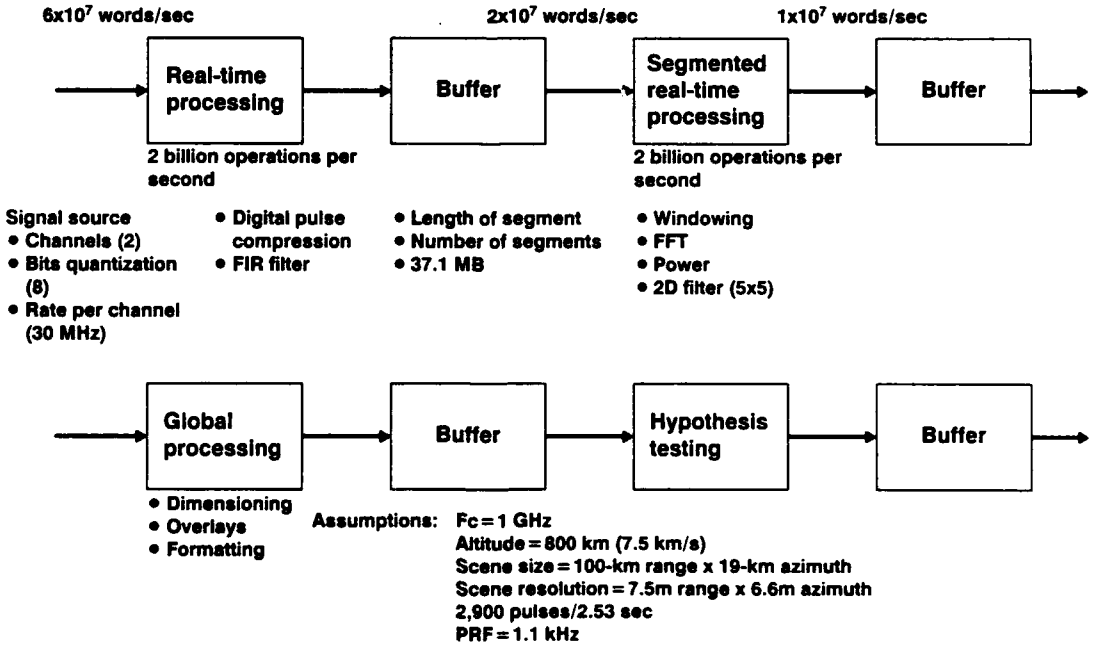


Figure 66
Typical SAR Processor

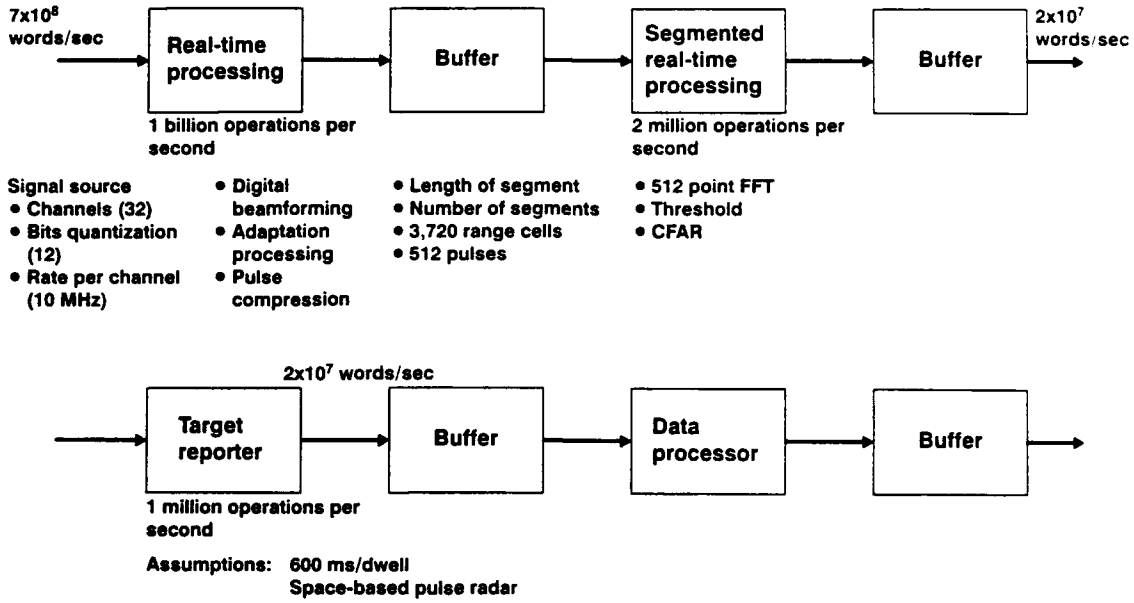


Figure 67
Typical SBR Concept

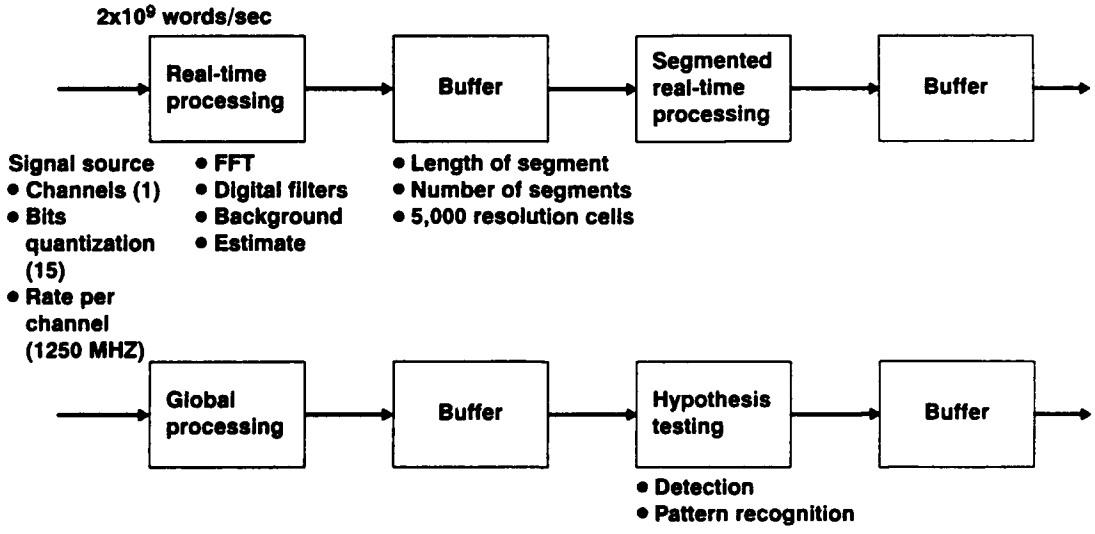


Figure 68
ESM Processor

8.3.2.4 Infrared Search and Track for AEW System. The AEW system aircraft is a radar surveillance aircraft that is used in many countries in the world today. The aircraft provides surveillance of airspace, and provides command and control and battle management for interceptors. One of the major upgrades currently under study is the addition of infrared sensors to the platform for infrared search and track capability. Sensor data fusion of infrared and radar data would take place aboard the platform for use in detecting missiles and cruise missiles.

One application for the augmented platform is the Theater Missile Defense (TMD) application. An engagement geometry is shown in the following figure (Figure 69). The geometry features a multiple interception capability for missile defense. To provide the capability, extensive signal processing would be accomplished on board the aircraft with new infrared signal processors. The development for such processors would be an excellent application for RASSP. In addition, the large platform would enable significant opportunity for use of the RASSP.

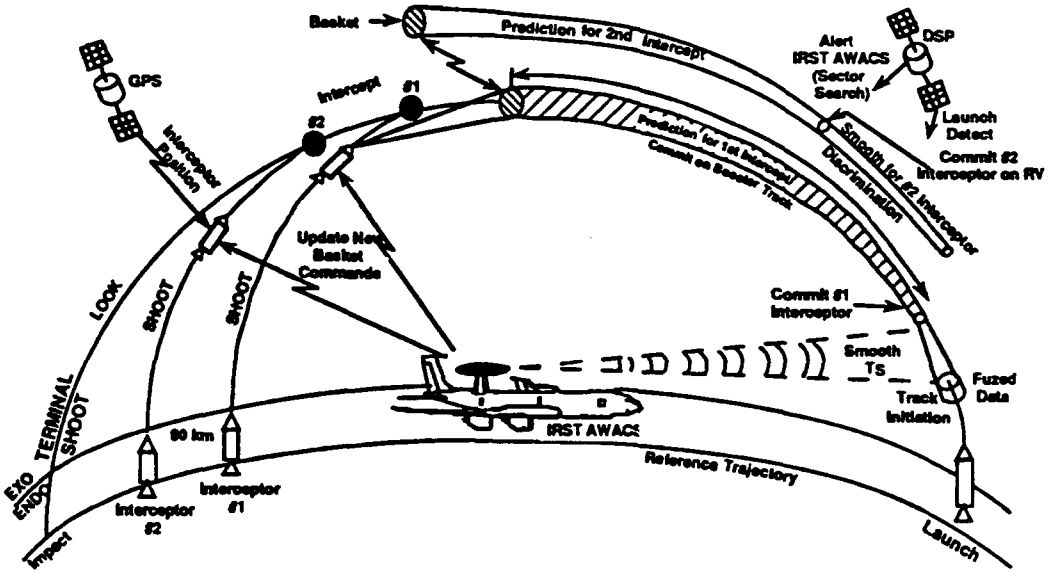


Figure 69
TMD Engagement Geometry

A top level decision tree for AEW system TMD application is shown in Figure 70. Several functions may be automated aboard the aircraft. Division of processing among the preprocessors, signal processors, and mission data processors would be dependent on electronics selected off-the-shelf and newly developed electronics. There are several options for RASSP implementation. Further, the range of possible AEW system roles in Theater Missile Defense is also extensive. Some options are shown in Figures 71 and 72.

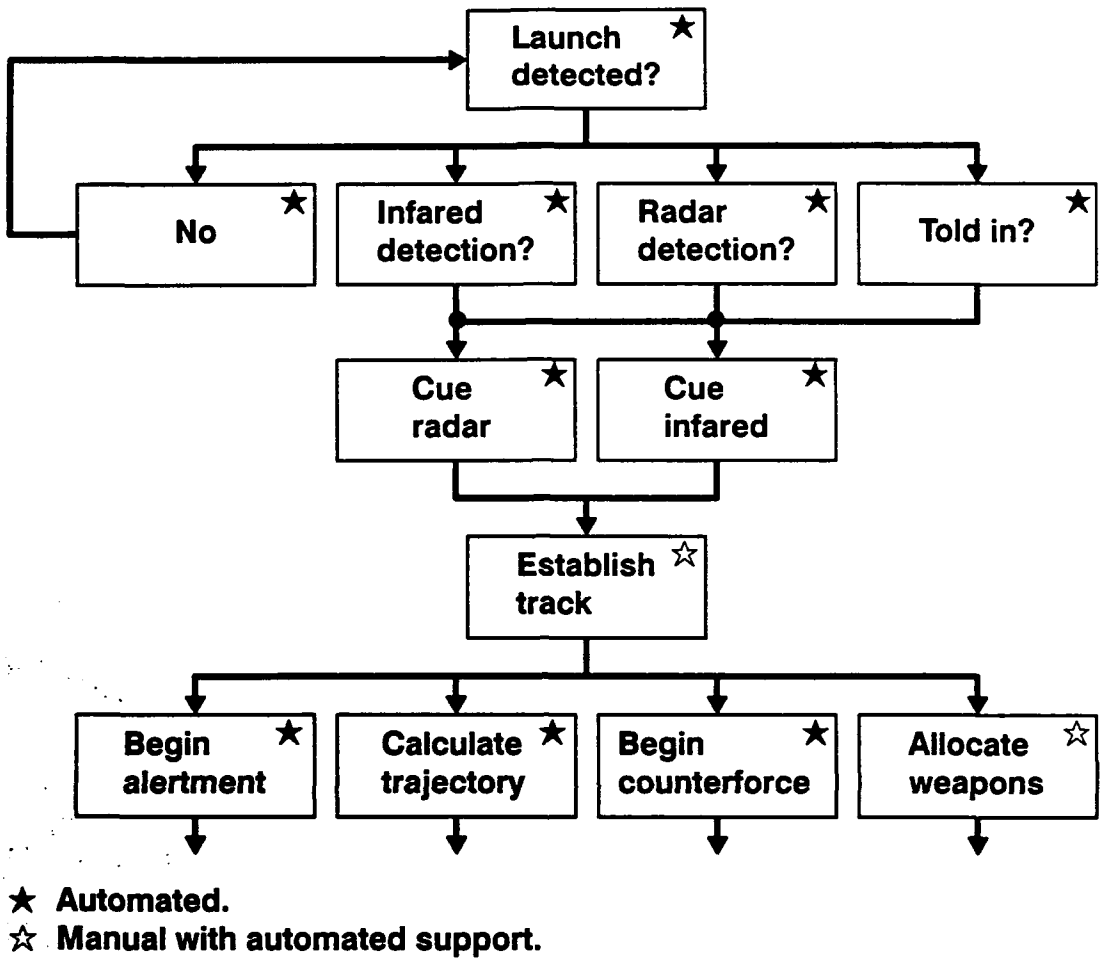


Figure 70
Top-Level Decision Tree for TMD

Signal processing for the infrared search and track advanced warning and control system platform consists of sensor data conditioning and preprocessing, time dependent processing, object dependent processing and object sighting messages transmission to the mission data processor on board. Representative processing rates and data handling rates are shown for a hypothetical engineering analysis. Signal processing rates vary from 2 gigops to 92 kops as data is processed. These rates would be very attractive for a good RASSP demonstration. The signal processing is shown in Figure 72.

Table 32 regarding AEW system processing application shows features associated with fusing of radar and IR data. In general, fusing at the pixel level provides the most attractive system level performance, but only at the cost of very complex, high performance signal processing electronics. At the other extreme, fusing of data at the target feature level greatly reduces requirements on the signal processing electronics, but at the cost of reduced system level performance. Intermediate regimes are shown in the Table 32.

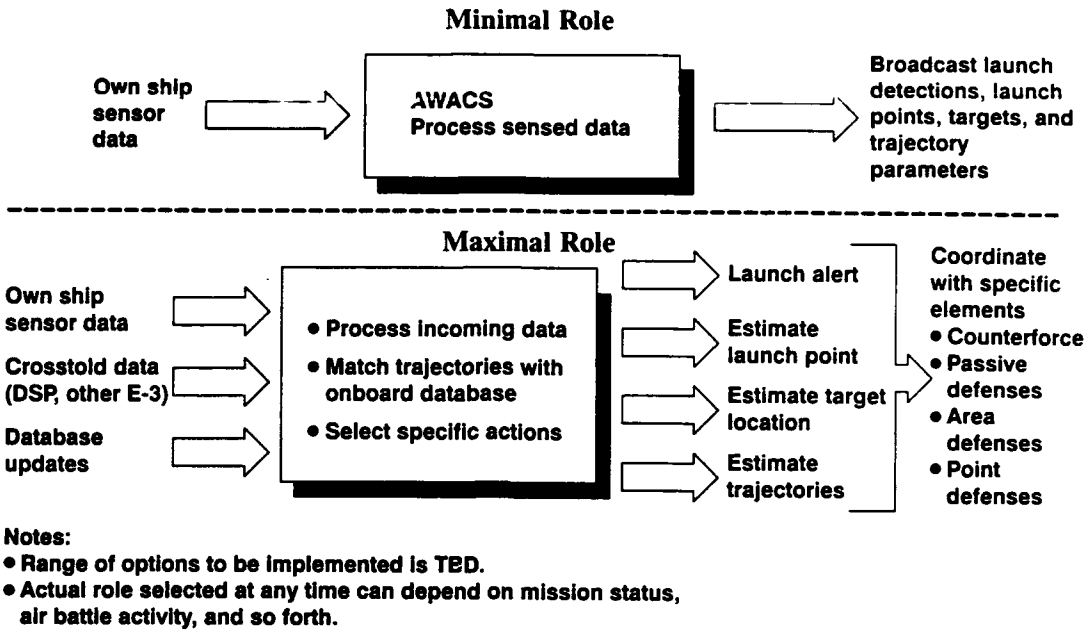


Figure 71
Range of Possible Roles in TMD

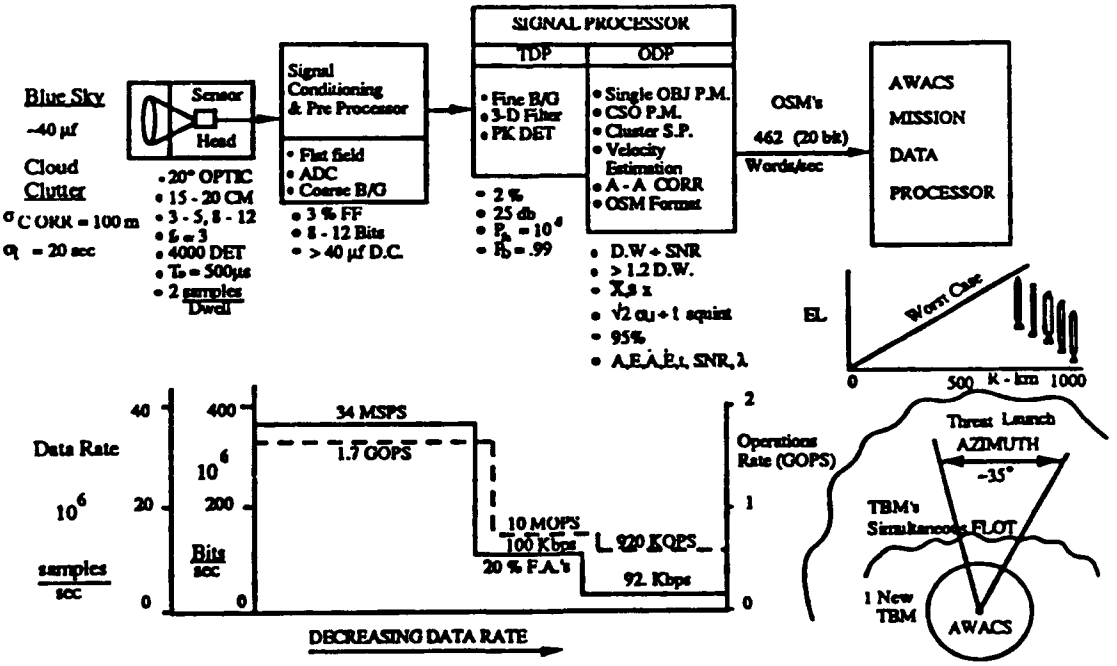


Figure 72
IRST Signal Processing

Table 32 Additional Airborne Platforms for RASSP Application

Pixel	Hit	Metric	Features
Raw detector outputs •10^{-5} pixels, 10 DPS •10 to 100 μrad pixel •0.4 to 0.7 μm and 3 to 20 μm	Threshold pixels at constant false alarm rate $< 10^{-4}$ data rate	•Sequence of hits (tracks) •Kalman filter	Derived discriminants (metric/radiometric)
•Highest data rate •Highest computation rate •Simplest algorithm	Data and computation rate dependent on false alarm and miss detection probability; P_f, P_m	Data and computation rate dependent on threat density, including closely spaced objects (CSO) < 2 pixels	•Data and computation rate driven by quality of penetration aids •Tank fragments to replica RV
Platform 1 Brilliant Eyes type •0.1 Gwords/sec •6 Gops (front TDP) •2D FIR •Visible/IR sensor	•$10^{-6} < P_f < 10^{-4}$ •$10^{-2} < P_m < 10^{-1}$ •7 x 7 packet for single objects •11 x 11 for clusters (dual CSO)	•AMA 200 μrad	•Optical cross section (E_A), ($\dot{E}_A$) •Color temperature, T, $\dot{T}$ •Radiant intensity, J, $\dot{J}$ •Derived x, y, z $\dot{x}$, $\dot{y}$, $\dot{z}$ $\ddot{x}$, $\ddot{y}$, $\ddot{z}$ (boost) •Launch point •Impact point
Platform 2 Airborne IRST •1.2 Gwords/sec •60 Gops •SATDI, 2D FIR filters	Similar requirements to platform 1	•Number of tracks/platform	Similar requirements with improved CSO raid count assessment (larger aperture)
Fused output algorithms •Pixel averaging	•Hit averaging	•Least squares error track merging	•Metric averaging •Radiometric match •Intelligence screening
Fusion feasibility (low 1995) Probable by 2000 to 2010	Fusion feasibility (low 1995) Probable by 2000 to 2010	Best bet by 1995	Best bet by 1993

9. SUMMARY

This section summarizes the key tasks for successful program execution and, for each task, provides an estimate of the percent of program technical effort, the level of program risk, and a discussion of risk mitigation.

9.1 Application Domain Selection
Effort: ~1% Risk: Low

The selection of the domain is important since it "bounds" the complexity of the hardware and software modules required to demonstrate the RASSP infrastructure. Effort and risk is low since the study has determined that several potential demonstration systems (domain instantiations) have suitable software and throughput requirements and well understood computational models.

9.2 Data Representation Standards Selection
Effort: ~5% Risk: Moderate

Selection of the standards to represent the functional, physical, and performance or parametric data that fully defines a hardware or software module at the system level, and provides for electronic data exchange with foundries that build components or modules, is of moderate risk since these standards are relatively immature in industry and will probably not be fully developed in the timeframe needed for the RASSP implementation phase. The risk can be mitigated by participation by all affected parties - CAx developers, module and signal processor developers, hardware foundries, the support software industry and the government customers - and a willingness to allow procedural overlays to immature standards to permit program execution to continue on schedule.

9.3 In-cycle Library Framework and CAx Tool Selection and Development
Effort: 30% Risk: High

The in-cycle or application design system includes selection or development of the simulators, emulators, and analysis/trade-off tools to perform the system design functions; selection of the framework and access method (interactive object modeler) for managing tool interaction with library element models; selection and integration of the requirements capture and traceability tools; and development of the proof of principle "test cases" and metrics for measuring tool effectiveness.

This task is high risk and requires significant effort since the existing tool base is immature and incomplete. VHDL languages and simulators have not been fully tested at this complex system level, emerging system analysis tools only model some of the required parametrics, and no standard modeling language exists that efficiently handles physical, functional, and performance attributes.

However, industrial and university research has demonstrated impressive, though limited, analysis tools with user-friendly graphical interfaces, and requirement capture and traceability tools are becoming available (such as ASCENT and SLATE). These capabilities are required to achieve RASSP cycle time and model year upgrade goals. This risk can be mitigated by management attention and focus on this task, and careful analysis of payoff versus risk when selecting these tools. For example, it is important that parametric models are complete for each hardware module, but it may be acceptable to manually input model parameters from the module design task if the risk of automatic model generation within the RASSP implementation phase time period is too high.

9.4 Hardware and Software Standards Selection
Effort: ~1% Risk: Low

Selection of the multiple standards for hardware interfaces, communication protocols, operating system functionality, etc., requires government and industry consensus, but the risk is low. Mature standards exist in the hardware interface and instruction set areas, and agreement is close on operating system standards. Risk of consensus and user acceptance of "open" signal processing module libraries increases if standards are over-specified.

9.5 Selection and Integration of the CAD/Case Libraries and Tools for Module Development
Effort: 15% Risk: Low to Moderate

The tools to design and validate out-of-cycle library elements are plentiful and will continue to improve without a RASSP focus. The task is to pick the best, integrate it into an open framework, and modify tool input/output to accept/provide the selection data representation standards. Considerable effort is required to evaluate, select and integrate the tools, but risk is low. Most signal processor software/hardware module builders have baseline systems in place today.

9.6 Select, Design, and Validate the Hardware/Software Modules to Support the Domain Specific Demonstration
Effort: 15% Risk: Low to Moderate

The effort and risk in the design of the library elements is manageable. Most builders of signal processors have candidate hardware building blocks in place today which can be reworked to fit the selected model requirements. Control of the complexity of the selected demonstration is key to a low risk effort for this task. Selecting and developing the software modules presents more risk, since software reuse rules are immature, and existing software for potential demonstration systems may not have been designed with appropriate modularity, thus requiring considerable rework.

9.7 Update and Demonstrate Seamless Interfaces to Flexible Foundries
Effort: 5% Risk: Low to Moderate

Flexible factories are coming on line quickly, pushed by commercial pressures and aided by government funding such as DARPA's MCM Foundry and MMST CIM efforts. However, some effort will be required to adapt foundry CAD to the RASSP defined interface standards and demonstrate the capability. Risk can move to the moderate range if we insist on overly aggressive manufacturing cycle time goals requiring large efforts and investment in immature CIM capital and software.

9.8 System Demonstration
Effort: 20% Risk: Moderate

Selection of the domain, and the instantiation of the domain to demonstrate, sets the risk for the system demonstration. Most of the effort should be in Task 9.6, described above. However, selection of an overly complex demonstration, and the potential delays and overruns that sometimes result, could negatively impact the perceived success of the program and inhibit fanout of the RASSP capability. Select demonstration candidates with well known computational models to mitigate this risk.

9.9 Technology Transfer
Effort: 5% Risk: Low to Moderate

The goal is completely open systems - available to all contractor and government organizations with a need for all or part of the libraries and tools. The risk is low if the total RASSP process is fully documented, including training programs, and the libraries and tools that support the process are fully documented and delivered, with support, to the user community by existing CAx suppliers. Since the integrated CAx system may include multiple CAx supplier tools, complete documentation of the integrated system should be required. Risk is managed by contract requirements for the above, and selection of a contractor team that can manage the diverse set of suppliers required to implement the RASSP system. Team leadership must include contractors with broad experience in signal processor development and manufacturing, supported, as required, by CAx developers and suppliers and modern MCM, SMT, and ASIC foundries.

APPENDIX A
CRITICAL ISSUES AND RECOMMENDATIONS

The following charts list several critical issues and recommendations identified during the execution of the RASSP study. These issues and recommendations are grouped by subject. The most important of these issues are discussed in detail in the body of the RASSP final report.

LIBRARY ISSUES/RECOMMENDATIONS

ISSUES	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
HW Interface Standards Local buses (inter-module)	Limit number of local bus types, but no single standard	Yes	Yes	JIAWG IEEE VME64	ONGOING	Select standards from interfaces currently available, for demo program
Test I/Fs (component level & module/external level)	Standardize on one component level I/F (JTAG), minimize module level & external I/F standards	Yes	Yes	JIAWG IEEE SAVA		
External I/Fs (Sensor I/Fs, Comm I/Fs)	Limited number of external I/F types, but no single standard. Comprehend wire, fiber optic, serial, and parallel.	Yes	Partial	JIAWG (HSDB), no standardization on high speed (sensor data) I/Fs		
SW Interface Standards Application/OS	Standardize interfaces between the OS and the application SW	Yes	Yes	POSIX	1994-1997	Use results of POSIX, augment for the demo as necessary.
OS Core/HW Drivers	Standardize interfaces between the OS core functions and hardware driver SW	Yes	No			Define standards for the RASSP demo
Library Development	Define and develop functional design (HW & SW) library elements for design reuse. Develop module level library designs for use in RASSP upgrades.	Yes	No			Develop initial library elements as part of the RASSP program. Include HW (Module & Functional) and SW (OS & Application) in library.

LIBRARY/DATA BASE ISSUES/RECOMMENDATIONS
HARMONIZATION NEEDED - OBJECT MODELS AND FRAMEWORKS

ISSUE	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Babel of non-interoperable object models (e.g. C++≠ PDES Express ≠ SQL3 ≠ OMG IDL, ETC)	Harmonization/convergence effort underway in ANSI X3H7	Highly desirable	more participation needed	ANSI X3H7 Object Model Interoperation	1992-1995	• Influence SQL3, participate in to X3H7 PDES Express • Fund R&D on Inter-object model sharing
Competing/overlapping frameworks efforts - OMG, PCTE, CFE, Pdes, EIS, ...Falcon, ...; need harmonization/convergence of object services and frameworks effort	• Harmonization/convergence effort just starting • Standard APIs and interchange formats	Highly desirable	More participation needed	OMG, PCTE, CFI, PDES/STEP, EIS, Sematech,...	Earliest-1995	• Support Workshop on Application Integration Architectures (11/92) • Sponsor similar harmonization work
Roadblock to seamless integration: Commercial CAD frameworks (e.g. Mentor Falcon) are proprietary/closed (may be technically open, not commercially); open systems frameworks (e.g. OMG) are immature.	Industry effort to develop open frameworks underway; use/influence emerging frameworks standards	Highly desirable	More participation needed	OMG, PCTE, CFI, PDES/STEP, EIS, Sematech, ...Mentor,...	Earliest 1995	• Influence CFI, CAX,...to build on OMG • Influence Mentor Falcon, DEC Powerframe,...to build on CFI • Develop standard API's & interchange formats • Insure CAD/frameworks are OODB independent (e.g. expenment-plug OODBs into Mentor Falcon, etc.)

LIBRARY/DATA BASE ISSUES/RECOMMENDATIONS
DATA BASE MANAGEMENT

ISSUE	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Standard 00DB API (interface) needed	Accelerate consensus leading to standards	Yes	Yes	ODMG DARPA Open 00DB X3H2 SQL3	1993-1995	• Encovrage ODMG consortion • Fund precompetitive consortion for 00DBs
Who will win-relational vs 00DB	Industry will develop hybrids; 00DBs provide improved functionality and performance; queries still needed.	Both	Yes	ODMG, DARPA Open 00DB, X3 00DB Task Group X3H2 SQL3	1993-1995	• Encovrage ODMG consortion • Fund precompetitive consortion for 00DBs
Federated and decentralized RASSP repository/ DBMS needed	• RASSP data stored in files, RDBs, 00DBs • Use dbms standards • Use design representation standards • Open federated dbms architecture • Open transaction protocol • Open query engine.	Yes	More work needed	X3H2.1 RDA, IBM DRDA, MMC cannot DARPA Persistent Object Base Program	1995	• Fund R&D on Enterprise-wide info. repositories; object file systems; distributed, WAV, and decentralized libraries; fine-grain change management, • Identify and real-time develop standards
Improvements to PDES/STEP Express	• Participation in PDES/STEP Express	Yes	Yes; More work needed			• Standard mapping to C++ • Queries for Express • Config. mgmt. for Express

LIBRARY/DATA BASE ISSUES/RECOMMENDATIONS
INTERCHANGE FORMATS AND LIBRARIES (GENERIC)

ISSUES	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Which interchange formats/domain representations to standardize; convergence of overlapping standards; inadequate standards.	• Consensus: HIGH: PDES/STEP, MEDIUM: rqmts spec. lang... LOW: ... • Focus on RASSP-specific interchange formats and representations	Yes	Yes--- More participation needed	CAD will it fit? Framework Initiative, Atlas	1994-1995	• Accelerate harmonization efforts • Use RASSP as a pilot program to accelerate CAD representation standards
What to put in the RASSP library?	• Any reusable component	Yes	No formal approach	DARPA STARS Repository, MCC Electronic Databook	1993-1995	• Influence DSP vendors and CAD vendors to publish CAD libraries • Prototype RASSP reuse library in pilot programs.
Bootstrap RASSP business	• RASSP libraries and services available to RASSP vendors • Amortize library reuse cost across organizations.	Yes	Yes--- More participation needed	ANSI X3T3 Open Distributed Processing, DARPA MADE, MADE, MCC EINet, OMG	1994-1995	• Fund several RASSP efforts which must cooperate • Demo RASSP trader/broker based on OMG/ODP/EINet • Fund development of PDES/STEP standards for RASSP

SYSTEMS ISSUES/RECOMMENDATIONS
SYSTEM DEFINITION AND SPECIFICATION

ISSUE	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Reusable SW architecture technology supporting RASSP	Extended DSSA	Yes	Yes	SEI DSSA	1995	Make formal part of RASSP; extend to accommodate DSHA
Reusable HW architecture technology supporting RASSP	Extended DSHA	Yes	No			New effort based on DSSA and RASSP
Ambiguous/imprecise specs; multiple spec & methodology aspects	Executable, visual specs; interrelated, hierarchical specs and methodologies	Yes	Yes	SEI SLATE Univs	1994-1996	IEEE standards efforts
Informal, imprecise activity model for HW and application designs	Formal process model for system design (integrated HW & SW design)	Yes	No		1995	Develop formal process model for system design, implementation, analysis, and experimentation
Maintenance of DSSA/DSHA Information Object Modelers	Develop formal "maintenance" process model	Yes	Partially	DSSA SEI Univs	1995	Must be extended to accommodate integration with DSHAs
Intelligent SW extraction from DSSA SW repository	Knowledge databases integrated into SW reuse	Yes	Yes	DSSA MARVEL		Adopt MARVEL-like design methodology
Historical use of HW design to guide/force SW design	Architectural synthesis based on executable specs	Yes	Yes	Univs	1997	Adopt standard, scalable, reconfigurable HW building block approach
Hard real-time vs. current trends in high-performance computer architecture	Deterministic/predictable HW execution models	Yes	Partially	Univs Workshops		On-going research subject; greater emphasis needed in architecture design efforts
Capture domain-specific HW and SW models into formal architectures	Architectural description language	Yes	Yes	DSSA SEI	1994	Rely on DSSA/SEI efforts
Multiple design and implementation goals	Decomposable, multiple architectural views	Yes	Yes			
Specification of real-time schedule						
Imprecise, informal model of "model year upgrade"	Develop formal "model year upgrade" process model(s)	Yes	No			Make integral part of RASSP
Ill-defined benchmarking activities for the system design process	formal process model for system design with defined measurement points and metrics	Yes	No			
Model year upgrade capabilities require increased levels of design automation	Formal process model for system design	Yes	Partially	Univs	1997	Combined descriptive and proscriptive, declarative process modeling language
Description of motion planning and control	Spatial operators algebra	Yes	Yes	JPL	1995	Track DSSA-related activities; extend to accommodate DSHA
Wide divergence of info needs and expertise by process model users	Visual representations, abstractions, and multiple perspectives	Yes	Yes	Univs	1997	
Reverse engineering of fielded systems for integration into DSSAs and DSHAs	Database translators, language translators, automatic partitioners, etc.	Yes	Partially	Univs Some corp.	1999	Make formal part of RASSP effort

SYSTEMS ISSUES/RECOMMENDATIONS
SYSTEM SIMULATION/EXECUTION

ISSUE	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Simulate detail vs. speed	Hybrid, hierarchical simulation	Yes	Yes	Univs UCB, et al	1995	Integrate into RASSP design system
Significant computational requirements of simulation	Massively parallel simulations	Yes	Yes	JPL, TI Jade ThinkM	1997	Parallel discrete-event simulation; parallel SPICE, parallel VHDL, etc.
Complexity of parallel computations	Parallel debuggers and visualizers	Yes	Yes	ONR Univs	Now	Need to be integrated into rest of RASSP design system
Decomposition of large Ada codes into parallel modules	Automatic SW partitioners Automatic parallelizers	Yes	Yes	Univs		Iterative compilation; human-assisted heuristics
Mapping SW to HW modules	Automatic SW mappers; dynamic load balancing	Yes	Yes	KAP ParaSoft Univs	Now-1997	Adapt to Ada; emphasis on simulated annealing and genetic algorithms
Reusable parallel algorithms	Scalable parallel libraries	Yes	Yes	DARPA (HPCC)	1997	Focus more on Dod domain application libraries
Scarcity of parallel trace tapes	Synthetic trace tapes; HW instrumentation for HPCC	Yes	Partially	Workshop Univs	1995	Develop tools for generating synthetic trace tapes
Multiple parallel computational models	Universal parallel computational model	No	Yes	Univs	1995	Follow NSF Grand Challenge effort
Mapping high-level modeling and simulation to CAD/CAE levels	Common/interoperable simulation languages; formal data interchange models	Yes	No			
Real-time schedule derivation	Off-line schedulers Dynamic schedulers	Yes	Yes	K. Schwan T. Baker K. Shin	1995	
Cost/performance trade-offs not easily understood/modeled	Virtual synthetic environments for visualizing complex data spaces	Yes	Yes	Univs		Greater emphasis on animated visualizations; virtual/synthetic world modeling technologies
Need design "advice" at multiple, interacting levels	Hierarchical, integrated design advisors	Yes	Yes	MCC CMU	1995	
Failure analysis difficult	Fault locators/analyzers coupled to design advisors	Yes	Partially	CMU		Back translators from logic to simulations to models
HW & system design cost models	Develop formal cost modeling language	Yes	Partially		1996	Extend COCOMO to accommodate synthesis of HW & system designs

SOFTWARE ISSUES/RECOMMENDATIONS
REUSE ISSUES

ISSUES	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Reuse vs. Performance	Demonstrate cost & cycle time benefit from software reuse vs. potential increased hardware costs	Yes	No		1993-1995	Perform cost/benefit analysis on RASSP demo program
Available reusable SW component libraries (models and code)	Define & Develop components for RASSP demo	Yes	No		1993-1996	Develop components as part of RASSP demo program
Domain Analysis	Define methodology and develop supporting environment	Yes	Yes	DARPA, CECOM	1993-1995	Monitor/leverage ongoing work into RASSP Domain Analysis
Reuse repository logistic issues (ownership, maint. responsibility, testing)	Define reuse methodology	Yes	Yes	DoD WG	1993-1995	
Reuseable component characterization	Define characteristics of component that must be captured	Yes	Yes	STARS	1993-1995	Monitor STARS progress, make recommendations based on RASSP demo program related progress

SOFTWARE ISSUES/RECOMMENDATIONS
SOFTWARE ENGINEERING ENVIRONMENT ISSUES

ISSUES	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Open data bases required to support information flow between tools	Vendors must define and abide by interface standards and data base access	Yes	Yes	Tool vendors	1993-1995	Continue requirements feedback to commercial efforts. Accelerates as necessary
Multiprocessor Target Debug Tools	Apply reuse methodology to develop scalable/reconfigurable debug tools; test I/F Stds	Yes	Yes	Tool vendors; JTAG	1993-1995	"
Configuration Management	Allow access to tools databases for CM across activities	Yes	Yes	Tool vendors	1993-1995	"
Number of HOLs supported	Use commercial language (C,C++) or get commercial support for Ada	Maybe	?	DoD		
Process/methodology/tools fanout. Change takes time and money	Define process. Feedback tool requirements to vendors	Yes	Yes	SEI, DoD contractors	1992-1996	
SEE capabilities dependent on host platform	Migrate to standard host platform. Allow vendors to concentrate efforts on product improvement rather than multiplatform support	Maybe	Yes	Tool vendors	1995	

HW CAD ISSUES/RECOMMENDATIONS

ISSUES	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Accelerated system simulation	Use of hardware accelerators for mixed mode design	Yes	Digital only	ZYCAD, IKOS	1993	Develop coherent mixed mode simulation strategy
Standards based data exchange	Industry concesus on language, schema, and data base level STDs	Yes	Yes	IEEE DASS, EDIF, IPO, CFI, OMG	1995	Support data harmonization
System level description languages	Extend language development to support concurrent engineering	Yes	No		1997-1998	Sponsor SLDL [illegible]. definition lang. harmonization, demos
Design advisors tradeoff analysis	Address at application and tech. domain level for disciplines	Yes	Yes	MCC (MCM level) universities	1994-1995	Develop demonstration suite of DA/TA tools for RASSP
Statistically based design	Six Sigma design, design of experiments, design centering	Yes	Limited	Six Sigma research institute	1995-1996	Applied research in stat. design approaches
DSP directed design and synthesis	Integrated toolsuite dev., *alogortihm to silicon*	Yes	Yes	Comdisco/Synopsis Mentor	1993	VHDL front ends, applied synthesis development

VHDL ISSUES/RECOMMENDATIONS

ISSUES	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Reusable signal proced. module library	Develop extensible set of VHDL modules for use in RASSP appl.	Yes	Limited	Protocal	1994-1995	Develop library as part of program
VHDL not optimized for DSP applications	Develop DSP targeted VHDL packages, silage HDL	No	Yes	Comdisco/Synopsis mentor		IEEE standardized DSP support package development
Proprietary data in vendor models	Develop data protection/cyphers into VHDL to protect vendor info	Yes	No			Develop compiled model scheme into RASSP program
Lack of timing analysis support	Develop path based worst case timing analysis, backannotation	No	Yes	Vital		Develop standardized timing format req for RASSP
VHDL based testability support	Synthesizable JTAG, BIST ATPG	Yes	Yes	Synopsis, Cadance, Mentor, others	1994-1995	Develop STD test feature requirements for RASSP
Behavioral level analog descriptions	MHDL, analog extensions for VHDL, analogy mast	No	Yes	Analogy, Mimic teams VHDL Groups (C30)	1994-1995	Analog extensions for VHDL prototype mixed mode anal. tools
Statistical modeling in VHDL	Allow parameter variances to be expressed in model	No	No		1997-1998	Develop statistical design models for RASSP STD VHDL Stat. packages

CALS ISSUES

ISSUES	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Vendors not supporting IGES Class III	Using VHDL and EDIF as workarounds	No	No			Accelerate release of PDES electrical API
Adaption of PDES by users/vendors	Vendors waiting on user requests users waiting on vendor releases F22 demonstration vehicle	No	Yes	IPO, PDES INC.		Sponsor support for PDES application
PDES/EDIF/CFI overlap	PDES/CFI/EDIF may share model views with different schemas	No	Yes	DEIF, PDES Group		Sponsor data harmonization migration paths into step
Development of Domain specific schemas	Design description requires to many model views	No	No			Develop us development and support centers for express
Unresolved API for step	Standard app procedural if for step	No	Yes	PDES Group		Identify RASSP relavent API

MANUFACTURING CAD ISSUES/RECOMMENDATIONS

ISSUE	APPROACH	MUST HAVE	BEING WORKED	WHEN	RECOMMENDATION
CAD-to-Manufacturing links #1	Harris supports "smart" GDSII, equipment vendors working with CAD vendors. IGES, PDES, WAVES, VHDL support in work.	Yes	Yes	1993-1994	Continued use custom interfaces until standards are supported.
Standard data exchange formats #2	ASEM Infrastructure projects addressing. Committees will approve.	Yes	Yes	1994-1995	Continue to use custom interfaces until standards are approved.
Design advisors #3	ASEM funding in place. MCC has initial version. Vendors are participating and will productize.	No	Yes	1994	Emulate design advisor and design packager and estimate productivity gains. Adapt ASEM software when available and measure.
Multipackage level CAD systems	Mentor, Cadence, Dasix integrating IC, MCM, and PWB CAD systems.	Yes	Yes	1994-1995	Use interfaces between packaging levels until S/W upgrades available.
IC, ASIC, MCM part models	Standards in work by committees.	Yes	Yes	1995	Continue to use currently defined models plus supplemental data.
RASSP Infrastructure	ASEM funding in place to address issues (consortiums, brokers,...)	No	Yes	1994-1995	Participate in consortiums: MCC, vendors, universities.
Real-time analysis (electrical/mechanical, high-performance)	Mentor Graphics, Cadence, Dasix, Harris pursuing in next generation CAD systems	Yes	Yes	1994-1995	Use standalone analysis in interim. Implement new CAD releases when available.
Global CAD databases	University development.. Vendor development in area as part of system integration efforts.	No	Yes	1995 +	Continue using and coordinating multiple databases. Use frameworks to aid in coordination.

NEXT GENERATION MANUFACTURING ISSUES

ISSUE	APPROACH	MUST HAVE	BEING WORKED	WHO	WHEN	RECOMMENDATION
Lower cost, manufacturing capability, cycle time	Process Improvement/technology	Yes	Yes	DARPA foundry programs DARPA ASEM program Mantech Foundry commercialization	1993-1995	Continued support of key technology contributors
Process modeling and simulation	Embed tools in next generation CIM system	Yes	Yes	MMST Program DARPA ASEM program	1996-1997	Extend capability to include key RASSP technologies
Customer access to foundry	Standard product descriptions and communications infrastructure	Yes	Yes	DARPA ASEM program Standards Committee(s)/MCC	1996-1997	DARPA sponsorship of efforts key to RASSP
Continuous Improvement	Closed loop manufacturing control	Yes	Yes	MMST program	1996-1997	Continued DARPA sponsorship of technology improvements
Translation of test from customer	Standardized test vector format, integrated with standard diagnostics, on high speed test equipment	Yes	Yes	Industry starting to address	1993-1994	Support of Industry standard vector format
Generation of manufacturing defects tests for MCM's	Automatic test pattern generation with & w/o boundary scan	Yes	No	Extension of current commercial software		Sponsor enhancement of commercial product
High pin count, high speed probes are expensive $8K $10K	Process Improvement development	Yes	No	Commercial probe vendors		Sponsor enhancement to assembly process and probe development

APPENDIX B
KEY STANDARDS

The following charts list several of the key standards and CAD related organizations, products, and efforts identified during the study.

RASSP RELATED WORK

COMPANY DOD PRODUCT STANDARDS ORGANIZATION	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY
CALS Initiative	CALS standards • IGES, EDIF, VHDL PDES/STEP,... SGML, ODA • IRDS, SQL, RDA, GOSIP, OSF NCS RPC, POSIX.1, X, EDI	Alignment with CALs stands
MCC	EINet Services	Broker RASSP libraries and services
Object Management Group (OMG)	• IDL/object model • CORBA Object Request Broker • Object services architecture	Open (non-proprietary) framework
Object Data Management Group	OODB Standards Proposal	OODB
CAD Framework Initiative (CFI)		eCAD framework standards
DARPA/TI	Open OODB	Reconfigurable OODB/framework tool kit
DARPA	MMST	Factory modeling
DARPA	DSSA	SW design
IBM	DSSA	SW design
CMU/Center for Dependable Computing	Micon PieScope Fiat IIE	Arch. Synth Visual Exec Fault Injection Instrumentation
STARS	Reuse process	Reuse
SEI/STARS	Process Definition Advisory Group	Reuse
SEI	Resident Affiliate Program	Reuse
SEI	Software Reuse Technology	Reuse
Cadre	Teamwork/OOA,OOD,ASG	SW Analysis & Design
Cadre	ASB, DSE	Ada code development
Cadre	Ensemble	C code development
Cadre	Architecture Design & Assessment System (ADAS)	HW/SW arch System/Component Simulation
Cadre	Teamwork/SIM	HW/SW arch System/Component Simulation
Cadre	Shortcut	Exec. Spec./Rapid Proto. of Req.
Cadre	Teamwork/Testcase	Testing
Cadre	Teamwork/RqT	Req. Mgmt.
Interleaf	TPS	Doc. Gen.
Software Systems Design	Ada/C Integrated Software Lifecycle Environment (ADADL based)	PDL/Code Analysis, Testing

RASSP RELATED WORK

COMPANY DOD PRODUCT STANDARDS ORGANIZATION	PRODUCT CAPABILITY STANDARD	RASSP APPLICABILITY
COMDISCO	Signal processing workstation	DSP Design
Intermetrics	VHDL/MHDL	Cad Tools
Mentor	Falcon Design Architect System 1076 Quicksim Quickfault Quickpath Quickgrade Accusim Packager Autologic DSPstation MCMstation Boardstation Systemstation	Framework Design Capture VHDL Design Digital SIM Timin Anal Fault Anal Fault Anal Analog SIM Man. IF Synthesis DSP Design MCM Design PWS Design Req. Cap
Synopsis	Synthesis Toolsuite	VHDL SIM/Synthesis

APPENDIX C
BIBLIOGRAPHY

1. Gass, w., Wong Y. et al, "Silicon Compilation of DSP Functions: Architecture and Circuit Technology"

2. IMEC90, "The Cathedral Silicon Compilation Environment"

3. Hartimo I., et al, "A Design Expert for Digital Signal Processing"

4. Wing J., "A Specifier's Introduction to Formal Methods"

5. Mandayam R., Vemuri R. "An Executable Notation for Performance Specification and Evaluation"

6. Mentor Graphics, Dataflow Language (DFL) User and Reference Manual

7. Lundberg L., "Generating VHDL for Simulation and Synthesis from a High Level DSP Design Tool"

8. Krishna A., Petrasko B. "Designing a Custom DSP Circuit Using VHDL"

9. Zhou W., "Investigation of Extension to a Digital Behavior-Level Hardware Description Language to Support Analog and Mixed-Mode Simulation"

APPENDIX D
REPORT DISTRIBUTION

(1) DARPA/ESTO
ATTN: Mr. Eliot D. Cohen
3701 North Fairfax Drive
Arlington, VA 22203-1714

(2) DARPA/OASB Library
3701 North Fairfax Drive
Arlington, VA 22203-1714

(3) Defense Technical Information Center
Building 5, Cameron Station
ATTN: Selections
Alexandria, VA 22304

(4) U.S. Army LABCOM
ET&DL
ATTN: Dr. C.G. Thornton, Director
SLCET-D
Fort Monmouth, JN 07703-5601

(5) Dr. Ingham Mack
Technical Area Manager for Electronic Devices
Office of Naval Technology
Code 224
800 N. Quincy Street
Arlington, VA 22217-5000

(6) WL/EL
ATTN: Mr. William J. Edwards
Wright-Patterson AFB, OH 45433-6543

(7) OUSDA Advisory Group on Electron Devices
c/o Palisades Institute for Research Studies
One Crystal Park
2011 Crystal Drive, Suite 307
Arlington, VA 22202

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	09 October 1992	Final Report, 05/12/92 - 10/12/92

4. TITLE AND SUBTITLE
CLIN 0002AB Final Technical Report, Rapid Prototyping of Application Specific Signal Processors Program

5. FUNDING NUMBERS
ACR: AA 9720400 1320
9219 PsH20 2525
DPAC2 5266 S49447
$156,208

6. AUTHOR(S) Dennis Best, Reagan Branstetter, David Counts, Dr. Doug DeGroot, Mark Eskew, Angela Harper, Dr. John Linn, Neal Stollon, Dr. Craig Thompson (all of Texas Instruments Incorporated)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Texas Instruments Incorporated
Integrated Circuits & Computers Department
P.O. Box 869305, M/S 8435
Plano, Texas 75086
6550 Chase Oaks Blvd., 75023

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
Electronic Systems Technology Office
Rapid Prototyping of Application Specific Signal Processors Program ARPA Order No. 9219/.4
Issued by DARPA/CMO under Contract #MDA972-92-C-0060

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency of the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A. Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)* This report discusses the results of the TI RASSP study efforts to identify a development process consistent with the RASSP program methodology. In this study, we defined the concept of a library based system design approach with in-cycle and out-of-cycle development of library elements. This report discusses this approach and the issues associated with system, hardware, and software design tools, and manufacturing foundry interfaces required to support a library based system development approach. The library based development concept is defined and the modularity and partitioning of a domain specific architecture is discussed. A description of the library elements supporting a library based system design approach is provided. Requirements, issues, and approaches to signal processing system design, analyses and tradeoffs, and system simulation and validation are discussed. Data base representation issues, framewords, and interfaces required to provide a seamless RASSP development environment are identified. Reusable software and library element development issues are addressed. Hardware CAD issues associated with library element development, system analyses, simulation, and validation are discussed. Manufacturing foundry CAD, CIM, design tool interfaces, and test issues are also discussed. A summary of the effort and recommendations for implementation are also included.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	167
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL